

**EFFICIENT SOLUTIONS TO TOEPLITZ-STRUCTURED LINEAR
SYSTEMS FOR SIGNAL PROCESSING**

A Dissertation
Presented to
The Academic Faculty

by

Christopher Kowalczyk Turnes

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering



Georgia Institute of Technology
May 2014

Copyright © 2014 by Christopher Kowalczyk Turnes

EFFICIENT SOLUTIONS TO TOEPLITZ-STRUCTURED LINEAR SYSTEMS FOR SIGNAL PROCESSING

Approved by:

Dr. James McClellan, Committee Chair
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Justin Romberg, Advisor
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Monson Hayes
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Jack Poulson
School of Computational Science and
Engineering
Georgia Institute of Technology

Dr. Chris Barnes
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Doru Balcan
Quantitative Finance
Bank of America

Date Approved: May 2014

Effort is one of the things that gives meaning to life. Effort means you care about something, that something is important to you and you are willing to work for it. It would be an impoverished existence if you were not willing to value things and commit yourself to working toward them.

– CAROL S. DWECK, *SELF-THEORIES*

To my wonderful, amazing, and patient parents Cynthia and Patrick, without whom none of this would be possible. Thank you for everything (but mostly for life).

ACKNOWLEDGEMENTS

Foremost, I wish to express my sincere gratitude to my advisor, Dr. Justin Romberg, for his support of my work and for his patience, time, and immense knowledge. Under his direction, my technical and communication skills improved dramatically. During my time at Georgia Tech, he did a phenomenal job of cultivating an amiable and social research group and achieved the impossible: making graduate school *fun*.

My research would not have been possible without the efforts of Dr. Doru Balcan. Doru took time out of a busy schedule to discuss my work, and often helped me to fully realize the significance and implications of my results. He has been a true mentor and friend.

I also wish to thank my other thesis committee members, Drs. Jim McClellan, Monty Hayes, Jack Poulson, and Chris Barnes, for their time and insight.

Many thanks to my brothers Jonathan and Walter for mastering that delicate balance between antagonism and encouragement over the course of my life, and to the rest of my extended family for their love and support.

I would also like to acknowledge several friends, colleagues, and labmates. To Cathy Zanetti, Laura Hansen, Peter and Alex Tuuk, and Peter Siy, the camping, parties, barbecues, pizza festivals, and innumerable other social events have made my time in Atlanta unforgettable. To Steve Conover, I will never forget our collaborations on coding ventures, as well as the many sushi and movie nights with Jiun-Hong Lai. To Adam Charles and Diane Isaacson, Sam and Taylor Shapero, Becky Fong, Kyle and Steph Krueger, Sean Kelly, Jeff Bingham, Jeff Dugger, and the rest of team “Shake and Bake,” thank you for joining me on so many trivia nights. To all of the coordinators and participants of the 12-hour “outside-the-box creative research initiatives,” thank you for suffering along with me. Lastly, to the many, many others who have served as friends, labmates, collaborators, and

hockey teammates, I thank you for all of the wonderful memories.

Finally, I am forever indebted to my wife and best friend, Auréle, without whose love and encouragement I would not have finished this dissertation. *Merci, mon amour.*

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	v
LIST OF TABLES	x
LIST OF FIGURES	xi
NOMENCLATURE	xii
SUMMARY	xiv
I INTRODUCTION AND FUNDAMENTALS	1
1.1 Introduction	1
1.2 Structured linear algebra	1
1.2.1 Types of structure	2
1.2.2 Structure in signal processing	6
1.3 A short history of Toeplitz algorithms	8
1.4 Displacement	12
1.4.1 Motivation and history	12
1.4.2 Sylvester and Stein displacement	13
1.4.3 Singularity and recoverability	15
1.4.4 Recovery formulas	17
1.4.5 Displacement operator matrices	18
1.5 Multi-level linear algebra	19
1.5.1 Basic definitions	19
1.5.2 Symmetry and persymmetry	22
II CHARACTERIZATION OF STRUCTURED SYSTEMS	27
2.1 Displacement and generators of structured matrices	27
2.1.1 Scalar Toeplitz matrices and inverses	27
2.1.2 Generalized Cauchy matrices and inverses	37
2.2 Polynomial interpolation and structured systems	42
2.2.1 Generalized polynomial interpolation	42

2.2.2	Structured systems and tangential interpolation	44
2.2.3	Solving tangential-interpolation problems	51
2.3	Schur recursions	60
2.3.1	Schur recursions for Toeplitz inverses	61
2.3.2	Schur recursions for generalized Cauchy inverses	62
III	NEW ALGORITHMS FOR SCALAR STRUCTURED MATRICES	67
3.1	Regularized least squares for Toeplitz systems	67
3.1.1	Tikhonov regularization	67
3.1.2	Regularization algorithm	69
3.1.3	Displacement and inverse generation	72
3.1.4	Implementation issues	75
3.1.5	Numerical results	82
3.2	Nonuniform resampling of digital signals	96
3.2.1	Structured matrices in non-uniform resampling	98
3.2.2	Superfast non-uniform-to-uniform resampling	100
3.2.3	Efficient pseudoinversion for resampling matrices	102
3.2.4	Performance scaling	105
3.2.5	Applications of interest	111
IV	MULTI-LEVEL TOEPLITZ THEORY AND ALGORITHMS	115
4.1	Inversion for special classes	115
4.1.1	Commutative classes of Toeplitz matrices	115
4.1.2	Inversion formulas and algorithms	117
4.1.3	Numerical results	124
4.2	One-level formulas for two-level Toeplitz inverses	128
4.2.1	Block-level displacement	129
4.2.2	Singularity of block-level displacement	132
4.2.3	Inversion through block-level displacement	133
4.2.4	Equivalence of block-level displacements	136

4.2.5	Extension to blockwise results	138
4.3	Miscellaneous results on two-level Toeplitz inverses	138
4.3.1	On two-level displacement and displacement rank	139
4.3.2	On two-level Gohberg-Semencul formulas	144
4.3.3	On two-level Toeplitz inverse generators	151
4.3.4	On transformation into two-level Loewner matrices	156
V	CONCLUSIONS	164
5.1	Summary of results	164
5.2	Future directions	166
APPENDIX A	— PROOFS	169
APPENDIX B	— TANGENTIAL-INTERPOLATION EXAMPLE	202
APPENDIX C	— SPIRALFFT	206
REFERENCES	231

LIST OF TABLES

1	Tan. int. accuracy	87
2	Tan. int. accuracy of artificial variables	88
3	Tan. int. accuracy for multi-column inputs	90
4	Tan. int. CG execution time comparison	92
5	Tan. int. CG error comparison	92
6	Tan. int. CG execution time comparison for multi-column inputs	93
7	Tan. int. CG accuracy comparison for multi-column inputs	93
8	SCTLT inversion deblurring accuracy	126
9	Equivalences for potential two-level inverse formulas	151
10	Common entries in two-level generators	154
11	SpiralFFT angle choice comparisons	217
12	SpiralFFT performance comparison	221
13	SpiralFFT volume reconstructions	229

LIST OF FIGURES

1	Example of circulant extension	46
2	τ -degree illustration	53
3	Tikhonov basis construction subdivision strategy	77
4	Comparison of number of conditions for extension strategies	81
5	Comparison of two new extension strategies	83
6	Tan. int. execution times	85
7	Tan. int. execution times for multi-column inputs	89
8	Tan. int. NUFFT reconstruction	96
9	Tan. int. NUFFT reconstruction error comparison	97
10	Resampling algorithm execution time scaling	109
11	Resampling algorithm error plots	110
12	Resampling algorithm NUFFT errors	113
13	Blurring filter for the image deblurring experiments	125
14	Deblurred images using the SCTLT inversion algorithm	125
15	Execution times for the SCTLT inversion algorithm (1)	127
16	Execution times for the SCTLT inversion algorithm (2)	127
17	Low-rank partitioning of the two-level Loewner form	162
18	Example spirals for SpiraFFT	211
19	SpiralFFT angle selection	216
20	SpiralFFT performance comparison	224
21	SpiralFFT performance curves	225
22	Synthetic and real 3-D volumes	226
23	3-D MRI reconstructions	228
24	Synthetic 3-D MRI reconstructions	228

NOMENCLATURE

F	Fourier matrix with entries $\mathbf{F}_{k,\ell} = \mathbf{e}^{-j2\pi k\ell/n}$ [Sec. 1.2.1.2].
diag (a)	Diagonal matrix formed from the entries of the vector a [Sec. 1.2.1.2].
\otimes_c	Circular convolution [Sec. 1.2.1.2].
\otimes	Linear convolution [Sec. 1.2.2.1].
$r_{xy}[n]$	Cross correlation of vectors x and y [Sec. 1.2.2.2].
$\nabla_{A,B}(\cdot)$	Sylvester displacement operator [Sec. 1.4.2].
$\Delta_{A,B}(\cdot)$	Stein displacement operator [Sec. 1.4.2].
rank (\cdot)	Matrix rank [Sec. 1.4.2].
I	Identity matrix (size assumed from context) [Sec. 1.4.3].
vec (\cdot)	Column-based vectorization of a matrix [Sec. 1.4.5].
$\Pi_{A,B}$	Sylvester displacement operator matrix [Sec. 1.4.5].
$\Theta_{A,B}$	Stein displacement operator matrix [Sec. 1.4.5].
\otimes	Kronecker product [Sec. 1.4.5].
$A^{\mathcal{T}}$	Block-level transpose of A [Sec. 1.5.1].
$A^{\mathcal{F}}$	Blockwise transpose of A [Sec. 1.5.1].
J_n	$n \times n$ exchange matrix [Sec. 1.5.2].
$A^{\mathcal{P}}$	Persymmetric transpose of A [Sec. 1.5.2].
$\mathcal{J}_{m,n}$	Block-level exchange matrix [Sec. 1.5.2].
$A^{\mathcal{P}}$	Block-level persymmetric transpose of A [Sec. 1.5.2].
$\mathcal{J}_{m,n}$	Blockwise exchange matrix [Sec. 1.5.2].
$A^{\mathcal{P}}$	Blockwise persymmetric transpose of A [Sec. 1.5.2].
Z_f	f -shift matrix [Sec. 2.1.1].
det (A)	Determinant of the matrix A [Sec. 2.1.1.3].
$\mathcal{G}(GDH^T)$	(Scalar) Gohberg-Semencul formula [Sec. 2.1.1.4].
$\widetilde{\nabla}^{-1}$	Inverse of a displacement operator [Sec. 2.1.2.2].

$\widetilde{\nabla}^\dagger$	Pseudoinverse of a displacement operator [Sec. 2.1.2.3].
$V_{1:n}$	Submatrix formed from first n columns of V [Sec. 2.2.1].
$\mathbf{p}(z)$	Vector polynomial [Sec. 2.2.1].
$C_k(T)$	Circulant extension of the matrix T [Sec. 2.2.2.1].
$\mathbf{circ}_k(T)$	Circulant matrix containing T as a submatrix [Sec. 2.2.2.1].
\mathbf{I}_k	Identity matrix of size $k \times k$ [Sec. 2.2.2.1].
$\mathbb{C}[z]^{d \times 1}$	Module of $d \times 1$ vector polynomials [Sec. 2.2.3.1].
$(\mathcal{R}(\mathbf{q}))_k$	Tangential-interpolation residual [Sec. 2.2.3.1].
$\tau\text{-deg}(\mathbf{p})$	τ -degree of the vector polynomial $\mathbf{p}(z)$ [Sec. 2.2.3.1].
\bar{a}	Complex conjugate of a [Sec. 3.1.3].
$\mathbf{sinc}(t)$	Sinc function $\sin(\pi t)/(\pi t)$ [Sec. 3.2].
$(\mathcal{S}, +, *)$	Commutative ring [Sec. 4.1.1].
$\mathbf{L}(Q)$	Block lower-triangular Toeplitz matrix [Sec. 4.2.3].
$\mathbf{U}(Q)$	Block upper-triangular Toeplitz matrix [Sec. 4.2.3].
$\mathfrak{G}_B(\cdot)$	Gohberg-Heinig formula [Sec. 4.2.3].
$\mathcal{D}(\cdot)$	Block diagonal operator [Sec. 4.3.4.1].
DFT	Discrete Fourier Transform.
FFT	Fast Fourier Transform.
FMM	Fast Multipole Method.
DMM	Diagonal-matrix multiply.
SSD	Structured sum decomposition.
ET	Extension and transformation.
CRTF	Complete recursive triangular factorization.
GKO	Gohberg-Kailath-Olshevsky.
LS	Least-squares.
C-L	Circulant-times-lower-triangular-Toeplitz product.
CG	Conjugate gradients.
SCTLT	Semi-commutative two-level Toeplitz.
PLS	Principal leading submatrix.

SUMMARY

This thesis presents a series of innovations in scalar (one-level) and multi-level structured linear algebra. In particular, a number of theoretical results are given that yield efficient algorithms for Toeplitz and Cauchy-like systems. While there are many types of structured systems, these two in particular are common to signal-processing applications. As a result, the algorithms developed in this research serve the purpose of reducing the computational burden of solving linear systems in real-world, practical problems.

This work is built upon a strong foundation of structured linear algebra that has grown steadily over the last several decades. While many concepts from this context have served as inspiration, one in particular – matrix displacement – is fundamental to most of the results in this thesis. In broad terms, the displacement of a given matrix structure concentrates the information characterizing a matrix into a compact form that may then be used to perform matrix multiplication in many fewer operations than usual. More importantly, displacement can also often be used to derive decompositions of both a matrix and its generalized inverse into a sum of structured multiplicative terms. These “structured-sum decompositions” (SSDs) allow for asymptotically efficient solutions to structured linear systems.

The major contributions of the research can be split into two categories.

Scalar results: Two efficient algorithms are given for scalar structured systems. The first is an algorithm for the solution to Toeplitz Tikhonov-regularized least-squares problems that expresses the regularization as a specialized polynomial interpolation. The interpolation problem is then solved with an existing algorithmic framework with superfast ($O(n \log^2 n)$) complexity. The second uses a basic trigonometric manipulation to express the sinc interpolation matrix involved in digital resampling as a generalized Cauchy matrix. A novel

superfast algorithm is then presented for calculating the pseudoinverse of the resampling matrix using an SSD.

Multi-level results: The second portion of the thesis presents a series of results on multi-level Toeplitz structure, divided into three parts. The first part describes an extension of scalar algorithms for Toeplitz systems to specialized classes of multi-level Toeplitz matrices by exploiting algebraic properties. Using these properties, existing superfast Toeplitz inversion algorithms are adapted for this class of matrices. The second part gives a theoretical contribution, providing a class of SSDs for the two-level Toeplitz inverse that exploits a single level of Toeplitz structure. These developments unify a series of scattered decompositions of the two-level Toeplitz inverse and give insight into the inverse structure of generic two-level Toeplitz matrices. Finally, the third part is a collection of properties of the multi-level Toeplitz inverse and a discussion of the difficulties that arise in extending scalar results to generic two-level matrices. Included in this part are discussions on multi-level displacement, potential SSDs of multi-level matrices, and transformations of multi-level Toeplitz matrices into related multi-level structured forms.

CHAPTER I: INTRODUCTION AND FUNDAMENTALS

1.1 Introduction

The field of signal processing is at the junction of engineering, mathematics, and computer science. Within signal-processing applications, a variety of complicated and nuanced problems are expressed as linear systems to which the algorithms of numerical linear algebra can be applied. Since the systems of these problems usually correspond to some physical quantities, it is often the case that their matrices are highly structured. One pervasive example is the Toeplitz structure that arises in temporally- or spatially-invariant systems.

There is a rich set of algorithms that have been developed to solve Toeplitz and other structured systems economically. While some of these algorithms have seen great popularity, their use is limited to one-dimensional problems. In multi-dimensional problems, structure appears in multiple ways or in multiple “levels” of the system matrix, yielding what are known as multi-level structured systems. With some exceptions, multi-level structured matrices lack many of the algebraic properties that are exploited to yield efficient scalar algorithms. This problem is especially pronounced for Toeplitz structure, where it has proven difficult to adapt existing superfast scalar algorithms for multi-level Toeplitz systems.

This research presents several asymptotically-efficient methods for structured systems. These algorithms are all connected to Toeplitz structure in some way; most are designed for Toeplitz and multi-level Toeplitz matrices, but even the generalized Cauchy pseudoinversion of Section 3.2 features Toeplitz structure prominently. To explain and derive these algorithms, however, it is first necessary to define several basic concepts and explore known results and techniques of structured linear algebra.

1.2 Structured linear algebra

By the early 1900s, the standard tools of linear algebra – Gaussian elimination, *LDU* factorization, *etc.* – were well-known methods of solving linear systems. These algorithms

are designed for arbitrary matrices, and a considerable amount of effort has gone into their study and implementation. However, these tools are intentionally broad in scope, and make minimal assumptions about the nature of the systems they solve.

This characteristic can be viewed as a deficiency for structured systems, as structure tends to dramatically reduce the number of free parameters defining a matrix. The primary tenet of structured linear algebra, which seems a natural conclusion to draw, is that the computational effort expended to solve a linear system should primarily be a function of the number of *free parameters in the system*, rather than the actual size of the system. More simply put, given an easily identifiable structure that reduces the number of parameters defining a matrix, it is a reasonable assumption that there should exist a way to exploit that structure to reduce the asymptotic cost of solving a linear system.

1.2.1 Types of structure

This research is concerned with a few specific types of structured matrices. The foremost focus is Toeplitz structure, which is abundant in signal processing applications. Of particular interest is multi-level Toeplitz structure, which arises in multi-dimensional problems. Closely related to Toeplitz structure (and also common to the field) is circulant structure, which has close ties to the Fourier transform. Cauchy structure also appears in one of the most basic signal processing problems – non-uniform resampling – and is of interest as well. Finally, Vandermonde structure is native to polynomial interpolation problems and similarly shares deep connections with the Fourier transform. The remainder of this subsection provides a brief introduction to each type of structure.

1.2.1.1 Toeplitz matrices

Temporally- or spatially-invariant linear systems are characterized by Toeplitz structure, and the matrices representing these systems have constant coefficient values along each diagonal. That is, a Toeplitz matrix $A \in \mathbb{C}^{m \times n}$ has coefficients $A_{i,j} = [a_{i-j}]$. Since it is entirely parametrized by its diagonal values, a generic Toeplitz system has only $(m + n - 1)$

degrees of freedom.

1.2.1.2 Circulant matrices

Circulant matrices are the subclass of square Toeplitz matrices that have periodicity in their coefficients. Specifically, an $n \times n$ circulant matrix $C \in \mathbb{C}^{n \times n}$ is a Toeplitz matrix $C = [c_{i-j}]$ such that $c_{-k} = c_{n-k}$ for $k = 1, \dots, n-1$, and is therefore parametrized by only n coefficients.

Circulant matrices are particularly appealing, as they are diagonalized by the Fourier matrix $\mathbf{F}_{k,\ell} = \frac{1}{\sqrt{n}} e^{-j2\pi k\ell/n}$. More specifically, if the first column of a circulant matrix C is $c = Ce_1$, then

$$C = \mathbf{F}^H \text{diag}(\sqrt{n}\mathbf{F}c) \mathbf{F}. \quad (1)$$

From this decomposition, it is evident that circulant matrices can be applied with a few Fast Fourier Transforms (FFTs) and diagonal matrix multiplies (DMMs) in $O(n \log n)$ operations.

The notion of circulant structure can be generalized to the class of f -circulant matrices.

Definition 1.1 (f -circulant matrix). *A matrix $A \in \mathbb{C}^{n \times n}$ is f -circulant if there are coefficients $a_i \in \mathbb{C}$ such that*

$$A = \begin{bmatrix} a_0 & fa_{n-1} & \cdots & fa_1 \\ a_1 & a_0 & \cdots & fa_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \cdots & a_0 \end{bmatrix}. \quad (2)$$

If $|f| = 1$, an f -circulant matrix can be diagonalized with Fourier-like operators as is shown in Section 4.1.1.

1.2.1.3 Cauchy, generalized Cauchy, and Loewner matrices

Cauchy matrices form a structured class whose elements are defined by $(m+n)$ parameters.

Definition 1.2 (Cauchy matrix). For $c \in \mathbb{C}^{m \times 1}$ and $d \in \mathbb{C}^{n \times 1}$ such that $c_i \neq d_j$ for all i, j , the matrix $C \in \mathbb{C}^{m \times n}$ with

$$C_{i,j} = \frac{1}{c_i - d_j} \quad (3)$$

is a **Cauchy matrix** parametrized by the nodes c and d .

The entries of the vectors c and d are referred to as the **denominator nodes**, and completely specify the matrix. Much like Toeplitz matrices, Cauchy matrices have a telescoping structure; any submatrix of a Cauchy matrix is also Cauchy. A well-known example of Cauchy matrices is the Hilbert matrix, whose coefficients have denominators

$$c_i - d_j = i + j - 1.$$

By the nature of their coefficients, Cauchy matrices can be highly sensitive to numerical errors. Consider the case where $c = \begin{bmatrix} 1.01 & 2.01 \end{bmatrix}^T$ and $d = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$. The Cauchy matrix

$$C = \begin{bmatrix} 100 & -\frac{100}{99} \\ \frac{100}{101} & 100 \end{bmatrix}$$

has singular values $\sigma_1 = 100.015$ and $\sigma_2 = 99.995$, for a condition number of 1.002. However, if the entry c_1 is set to $c_1 = 0.9999$ (a change of only 1%), the condition number rises to 100. For this reason, algorithms for Cauchy matrices are often heavily stabilized.

The structure of Cauchy matrices allows them to be applied efficiently. One option is to use polynomial interpretations of Cauchy matrix multiplication [83], but this can be highly numerically unstable depending on the denominator nodes. Alternatively, a Cauchy matrix can be *approximately* applied in $O(N)$, where $N = m + n$, with the Fast Multipole Method (FMM) [50].¹

Cauchy structure can be generalized to a broader class of matrices that incorporate numerator factors, as well.

¹The $O(N)$ complexity assumes a fixed precision; in reality, the execution time scaling is $O(N \log(1/\epsilon))$, where ϵ controls the precision.

Definition 1.3 (Generalized Cauchy matrix). For $c \in \mathbb{C}^{m \times 1}$ and $d \in \mathbb{C}^{n \times 1}$ such that $c_i \neq d_j$ for all i, j , and for $Z \in \mathbb{C}^{m \times r}$ and $Y \in \mathbb{C}^{n \times r}$ with rows z_i^T and y_j^T , respectively, the matrix $B \in \mathbb{C}^{m \times n}$ with entries

$$B_{i,j} = \frac{z_i^T y_j}{c_i - d_j} \quad (4)$$

is a **generalized Cauchy matrix** parametrized by the denominator nodes c_i and d_j and the generator matrices Z and Y .

Defining the Cauchy matrix $C_{i,j} = (c_i - d_j)^{-1}$, the generalized Cauchy matrix B has an SSD

$$B = \sum_{k=1}^r \text{diag}(Z_k) \cdot C \cdot \text{diag}(Y_k), \quad (5)$$

where $Z_k = Ze_k$ and $Y_k = Ye_k$ are the columns of Z and Y . Given this decomposition, the cost of applying a generalized Cauchy matrix is $O(rN)$ when the FMM is employed [83].

A subclass of generalized Cauchy matrices is the set of Loewner matrices.

Definition 1.4 (Loewner matrix). For $a, c \in \mathbb{C}^{m \times 1}$ and $b, d \in \mathbb{C}^{n \times 1}$ such that $c_i \neq d_j$ for all i, j , the matrix $B \in \mathbb{C}^{m \times n}$ with entries

$$B_{i,j} = \frac{a_i - b_j}{c_i - d_j} \quad (6)$$

is a **Loewner matrix** parametrized by denominator nodes c and d and numerator nodes a and b .

Loewner matrices are generalized Cauchy matrices for which $z_i^T = \begin{bmatrix} a_i & -1 \end{bmatrix}$ and $y_j^T = \begin{bmatrix} 1 & b_j \end{bmatrix}$. These matrices are closely connected with interpolation problems for rational functions (i.e., “rational interpolation”).

1.2.1.4 Vandermonde matrices

Vandermonde matrices correspond to polynomial evaluations, and are parametrized by a set of m “evaluation nodes” and the number of columns, n .

Definition 1.5 (Vandermonde matrix). For $z \in \mathbb{C}^m$, the matrix

$$V = \begin{bmatrix} 1 & z_1 & \cdots & z_1^{n-1} \\ 1 & z_2 & \cdots & z_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & z_m & \cdots & z_m^{n-1} \end{bmatrix}.$$

is a **Vandermonde matrix**.

Definition 1.5 is not universally agreed upon; depending on the application, either the matrix V or its transpose V^T may be referred to as “Vandermonde.”

When the evaluation nodes are the roots of unity $z_k = e^{\pm j2\pi k/m}$, V is a Fourier matrix and can be applied in $O(n \log n)$ operations with the FFT. When $m = n$ as well, the matrix V may also be inverted in $O(n \log n)$ with the FFT. When the nodes are more general, V can be applied to a vector in $C(n) \log n$ operations, where $C(n) = O(n \log n \log \log n)$, with the fan-in/fan-out method [83].

1.2.2 Structure in signal processing

Some of the most basic signal-processing operations and concepts are closely connected with structured systems. The following sections contain a representative – but highly incomplete – set of examples.

1.2.2.1 Convolution

Linear convolution is natively tied to Toeplitz matrices, and the convolution of two vectors can always be written as a linear system involving a Toeplitz matrix. If h and x are two 1-D signals, their linear convolution is written as $y = h \otimes x$. If the lengths of h and x are m and n , respectively, then y is of length $p = m + n - 1$, and it can be written as the left-hand side of a linear system $y = Tx$, where T is a $p \times n$ Toeplitz matrix with entries $T_{i,j} = h_{i-j+1}$.²

²It is assumed that $h_k = 0$ for $k \notin [0, m - 1]$.

The problem of deconvolving two signals is equivalent to solving a Toeplitz system. Any length- n subvector y_s of y can be expressed as $y_s = T_s x$, where T_s is the corresponding square submatrix of T . Assuming h to be known and T_s to be nonsingular, the signal x can be recovered from the measurement subvector y_s as $T_s^{-1} y_s = x^*$.

In the case of higher-dimensional signals, the previous explanations have straightforward generalizations. For 2-D signals, let h and x be images and $y = h \otimes x$ their 2-D linear convolution. Denoting the column-vectorized image x as $\dot{x} = \mathbf{vec}(x)$, then $\dot{y} = \mathbf{vec}(y) = T \dot{x}$, where T is a matrix with a Toeplitz arrangement of Toeplitz blocks $T_{i,j}$. The entries of each block are given by $(T_{i,j})_{k,\ell} = h_{i-j+1, k-\ell+1}$. This type of matrix is referred to as a **two-level Toeplitz** matrix.

Supposing x to be $m \times n$, an $m \times n$ submatrix y_s of the left-hand side y can be written (in vectorized form) as $\dot{y}_s = \mathbf{vec}(y_s) = T_s \dot{x}$, where T_s is an $mn \times mn$ submatrix of T . More specifically, T_s has an $n \times n$ Toeplitz arrangement of blocks, each of which is an $m \times m$ Toeplitz matrix, and is therefore a two-level Toeplitz matrix. If h is known and T_s is nonsingular, the vectorized input \dot{x} may be recovered from y_s by computing $\dot{x} = T_s^{-1} \dot{y}_s$. Further generalizations to higher dimensions are immediate.

In addition to linear convolution, circular convolution also involves structured matrix operations. A circulant matrix-vector product produces the result of a circulant convolution. That is, given $h, x \in \mathbb{C}^n$, the circular convolution $y = h \otimes_c x$ is the product

$$y = \mathbf{F}^H \text{diag}(\sqrt{n} \mathbf{F} h) \mathbf{F} x = \mathbf{F}^H \text{diag}(\sqrt{n} \mathbf{F} x) \mathbf{F} h.$$

If a circulant matrix is nonsingular, its inverse is easily computed from its factorization; given $C = \mathbf{F}^H \text{diag}(\sqrt{n} \mathbf{F} c) \mathbf{F}$, then $C^{-1} = \mathbf{F}^H \text{diag}(\sqrt{n} \mathbf{F} c)^{-1} \mathbf{F}$.

1.2.2.2 Autocorrelation and cross-correlation of random processes

The cross-correlation of two discrete signals is given by the sum

$$r_{xy}[n] = \sum_m \overline{x[n]} y[m+n].$$

Assuming x and y are both finite in time, this amounts to the linear system

$$\begin{bmatrix} r_{xy}[1-n] \\ r_{xy}[2-n] \\ \vdots \\ r_{xy}[0] \\ r_{xy}[1] \\ \vdots \\ r_{xy}[n-1] \end{bmatrix} = \begin{bmatrix} \overline{x[n-1]} & 0 & \cdots & 0 \\ \overline{x[n-2]} & \overline{x[n-1]} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \overline{x[0]} & \overline{x[1]} & \cdots & \overline{x[n-1]} \\ 0 & \overline{x[0]} & \cdots & \overline{x[n-2]} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \overline{x[0]} \end{bmatrix} \begin{bmatrix} y[0] \\ y[1] \\ \vdots \\ y[n-1] \end{bmatrix},$$

where x and y are both assumed to be of length n for simplicity. It is obvious from the expression above that the cross correlation between two discrete signals can be written as a Toeplitz matrix multiplication (and hence also as a convolution).

Similarly, the autocorrelation of a digital sequence $x[n]$ is the cross-correlation $r_{xx}[n]$. From the previous expression, the cross-correlation is an even function, and $r_{xx}[-k] = r_{xx}[k]$. The autocorrelation matrix $R_{i,j} = r_{xx}[i-j]$ of a wide-sense stationary process x is then a square, symmetric Toeplitz matrix.

1.2.2.3 Applications

While convolution and cross correlation refer to common problems in signal processing, there are many more *specific* applications where Toeplitz matrices are encountered, as well. Examples include integral equations with difference kernels [47], functional approximation [16], interpolation [92], time-series regression [28], and scattering theory [17]. Of course, these examples are but a small subset of the problems in which structure arises.

1.3 A short history of Toeplitz algorithms

Since Toeplitz matrices appear often in practical problems and their structure is readily visible, they have served as an obvious candidate for the development of efficient algorithms. In 1947, Norman Levinson was the first to develop an $O(n^2)$ recursive inversion

algorithm for Toeplitz matrices [70]. This algorithm exploited the telescoping structure of the principal leading submatrices of Toeplitz matrices, as well as their symmetry properties. Over the next two decades, Levinson’s work was re-derived and improved by Durbin [28], Trench [102], Zohar [116], and others. Each of these works operated in a unique context, but used the same fundamental idea.

In the late 1960s, Bareiss developed a recursive inversion algorithm more closely resembling the methods of classical linear algebra by using Toeplitz structure to accelerate the row reduction in *LDU* factorization and Gaussian elimination [7]. He used information from the upper- and lower-triangular row-reduced forms to bring the total cost of Gaussian elimination for Toeplitz systems to $O(n^2)$ calculations. Since Bareiss’s algorithm effectively performs *LDU* factorization, which progressively reduces the system with Schur complements, it is referred to as a **Schur recursion**.³ While the Schur recursion requires twice as much storage as the Levinson recursion, it involves no inner product calculations and can be considerably faster when implemented in parallel architectures.

The Levinson and Schur recursions were important developments and remain sufficient for moderately sized problems. However, the advent of Cooley and Tukey’s $O(n \log n)$ FFT in 1965 caused a radical change in the landscape of scientific computing [25]. The FFT allowed convolution and polynomial multiplications to be calculated significantly faster than prior techniques had allowed. As a result, the direct-form calculations of circular and linear convolutions in signal processing were replaced with FFT-based operations. In structured linear algebra, the FFT allowed Toeplitz and circulant matrix-vector products to be computed in $O(n \log n)$ time.

The rising prominence of the FFT led to a renewed interest in the Schur and Levinson recursions, resulting in the first set of “superfast” Toeplitz solvers. These algorithms drew heavily from the FFT; not only did they employ it to accelerate the calculations of the

³While not originally known, it was shown that this general technique had deep connections to Issai Schur’s prominent work on complex function theory, hence the association with Schur’s name [94].

recursions, but they also mimicked its divide-and-conquer approach. Regardless of the specifics, recursion remained the unifying theme for each of these algorithms.

The first superfast Toeplitz solver was presented in 1980 by Morf [77], and was centered on the displacement-rank approach introduced in [76] and later formalized in [63]. Morf's algorithm used the properties of the displacement structure of Toeplitz matrices along with a blockwise matrix-inversion formula to derive a divide-and-conquer variation of the Schur recursion. In the same year, Bitmead and Anderson developed a nearly identical procedure [15]. These two algorithms were the first to achieve a complexity less than $O(n^2)$, but suffered from large overhead cost and ultimately proved impractical [95].

Several years later, a second generation of accelerated recursions emerged that eschewed displacement for polynomial methods, beginning with Musicus in 1984 [80] and followed by a variety of similar algorithms [3, 4, 27]. By placing the recursions into a polynomial context, these algorithms used the FFT to accelerate polynomial multiplications and again took a divide-and-conquer approach to the inversion. Compared to the earlier displacement-based algorithms, these approaches had considerably smaller overhead costs, but were prone to numerical instability. Modern versions of these accelerated recursions improve the stability at the cost of a slightly higher complexity of $O(n \log^3 n)$ [99].

The accelerated recursions improved the asymptotic complexity of the fast algorithms, but failed to gain mass appeal. A new idea for superfast Toeplitz inversion algorithms surfaced several years later that involved a fundamentally different approach. The objective of this new method was to find the parameters defining the underlying structure of the Toeplitz inverse.

The Toeplitz inverse had been studied well before the accelerated recursions were introduced; its closed-form expression was known even prior to Durbin's work [97, 115]. However, this description of the Toeplitz inverse did not appear computationally useful until the derivation of the Gohberg-Semencul formulas in 1972 [48]. These formulas decompose the Toeplitz inverse as a sum of products of triangular Toeplitz matrices, meaning

it could be applied at the cost of a few Toeplitz matrix-vector products (using the FFT). Decompositions like the Gohberg-Semencul formulas, (1), and (5) that express a matrix as a sum of products of structured matrices are referred to as “structured-sum decompositions” (SSDs), and are a recurring theme in structured linear algebra.

For the Gohberg-Semencul formulas to be useful, however, there must exist an efficient way to determine their parameters, known as either the **canonical fundamental system (CFS)** or the **inverse generators** of the matrix. The first step toward a superfast algorithm to determine the inverse generators was to place them in the context of polynomials [56]. This point of view later allowed Van Barel to express the generators as solutions to a specialized type of problem known as **tangential interpolation** [110]. Using the proper theoretical framework [108] and employing the FFT for polynomial multiplications and evaluations, Van Barel gave a stabilized divide-and-conquer algorithm to produce the inverse generators of a Toeplitz matrix. When paired with the Gohberg-Semencul formulas, this result provides an $O(n \log^2 n)$ Toeplitz inversion algorithm. Compared to alternatives, the tangential-interpolation algorithm is more intricate, both in terms of theory and implementation. However, for very large matrices, it can provide extremely efficient inversion relative to alternatives.

Finally, more recent Toeplitz inversion algorithms make a subtler use of Toeplitz structure, employing efficient transformations to replace Toeplitz system of equations with Loewner or generalized Cauchy forms. The transformed systems have a highly compressible structure that allows them to be solved efficiently and stably with hierarchical techniques. With the transformed system solved, the original solution to the Toeplitz system is found by inverting the transforms.

Two prominent examples of this approach exist. The first, developed by Martinsson and Rokhlin [73], transforms the Toeplitz matrix with the 2-D Fourier transform. The off-diagonal blocks of the transformed matrix have low rank, a property shown in Rokhlin’s previous work on non-uniform Fourier transforms [30]. This rank structure is used to solve

the system in a fashion inspired by the FMM [20], yielding an inversion complexity of $O(n \log n)$.

The second algorithm to take this approach transforms a Toeplitz matrix into a Loewner matrix using two Fourier-like operators. This transformation was first given by Fiedler, who showed how Hankel matrices (which are column-reversed Toeplitz matrices) can be transformed into their Loewner forms [39]. The analogous result for Toeplitz structure surfaced in later works [44, 47, 52, 82].

The Loewner matrix resulting from the transform also has off-diagonal blocks with low rank. This property is exploited by expressing the Loewner form in a matrix decomposition called the “sequentially semi-separable” (SSS) representation [21]. Using this structure, Chandrasekaran developed a compressive inversion algorithm that operates in $O(np^2)$ time, where p is the maximum rank of any of the off-diagonal blocks (and can be shown to be small compared to n) [22].

1.4 Displacement

Many modern results in structured linear algebra make use of a technique known as the displacement-rank approach. This technique distills a structured matrix down to a set of fundamental parameters that characterize it. With a compact representation of a given matrix structure, many operations can be performed efficiently.

1.4.1 Motivation and history

The earliest origins of the displacement-rank approach appear to be in the context of radiative transfer problems [60], with its central ideas later resurfacing in the least-squares estimation of stochastic processes in terms of nonlinear Riccati equations [61]. However, in a structured linear algebra context, the displacement framework began with the specific intent of characterizing how “close to Toeplitz” a given matrix is [63]. In this context, these first displacement operators made use of the evident property that Toeplitz matrices define systems that are in some way shift invariant [40].

However, the use and definition of displacement rapidly expanded, and the focus of displacement operators broadened from Toeplitz matrices to structured matrices in general. Pan gives a more modern and encompassing definition for displacement operators [83]:

...the displacement of a matrix M [is] the image $L(M)$ of an appropriate linear displacement operator L applied to the matrix M and revealing its structure.

While the term “revealing its structure” is somewhat vague, the general idea is that the displacement yields some compact, meaningful description of the matrix. More specifically, it typically yields a low-rank structure whose total description requires as many parameters (or perhaps just a few more) than is required to parametrize the matrix.

The displacement-rank approach is useful when the low-rank descriptions lend themselves well to efficient algorithms. For many classes of structured matrices, displacement generates efficient multiplication and inversion algorithms in which the structure of the matrix is inherently exploited through the choice of displacement operator. In designing a displacement algorithm, then, it is of paramount importance to understand the theoretical and numerical properties of the operator.

1.4.2 Sylvester and Stein displacement

There are two types of displacement operator used for structured matrices.

Definition 1.6 (Sylvester displacement). *For matrices $C \in \mathbb{C}^{m \times n}$, $A \in \mathbb{C}^{m \times m}$, and $B \in \mathbb{C}^{n \times n}$, the **Sylvester displacement** of C by A and B is*

$$\nabla_{A,B}(C) = AC - CB. \quad (7)$$

The operator $\nabla_{A,B}(\cdot)$ is referred to as a **Sylvester displacement operator**. Sylvester displacement is a linear operation; given two matrices C and D , it follows from Definition 1.6 that $\nabla_{A,B}(C + D) = \nabla_{A,B}(C) + \nabla_{A,B}(D)$.

Fact 1.1 lists some other properties of Sylvester displacement.

Fact 1.1. For $A, B, C, D, E \in \mathbb{C}^{n \times n}$, the following are true:

1. $\nabla_{A,B}(CD) = \nabla_{A,E}(C)D + C\nabla_{E,B}(D)$ through direct evaluation.
2. $C^{-1}\nabla_{A,B}(C)C^{-1} = -\nabla_{B,A}(C^{-1})$ if C is nonsingular from Definition 1.6.
3. $\text{rank}(\nabla_{A,B}(C)) = \text{rank}(\nabla_{B,A}(C^{-1}))$ if C is nonsingular from the previous point.
4. if A, B , and C are symmetric, then $(\nabla_{A,B}(C))^T = -\nabla_{B,A}(C)$.
5. if A, B , and C are persymmetric, then $(\nabla_{A,B}(C))^P = -\nabla_{B,A}(C)$.

Given the displacement structure of a matrix, the second point of Fact 1.1 shows how the displacement structure of the inverse may be obtained. This property is crucial to several algorithms, and is a particularly beneficial aspect of Sylvester displacement.

A second type of displacement is Stein displacement.

Definition 1.7 (Stein displacement). For matrices $C \in \mathbb{C}^{m \times n}$, $A \in \mathbb{C}^{m \times m}$, and $B \in \mathbb{C}^{n \times n}$, the **Stein displacement** of C by A and B is

$$\Delta_{A,B}(C) = C - ACB. \quad (8)$$

The operator $\Delta_{A,B}(\cdot)$ is referred to as the **Stein displacement operator**. Stein displacement is also linear, since for two matrices C and D it is evident that $\Delta_{A,B}(C + D) = \Delta_{A,B}(C) + \Delta_{A,B}(D)$.

The following statements are also true of Stein displacement, and are analogous to several points of Fact 1.1.

Fact 1.2. For $A, B, C, D \in \mathbb{C}^{n \times n}$, the following are true:

1. *The relations*

$$\Delta_{A,B}(CD) = \frac{1}{2}\Delta_{A,I_n}(C)(2D - \Delta_{I_n,B}(D)) + \frac{1}{2}(2C - \Delta_{A,I_n}(C))\Delta_{I_n,B}(D),$$

$$\Delta_{A,B}(CD) = \Delta_{A,I_n}(C)D + AC\nabla_{I_n,B}(D), \quad \text{and}$$

$$\Delta_{A,B}(CD) = C\Delta_{I_n,B}(D) - \nabla_{A,I_n}(C)DB,$$

which are shown through direct evaluation as in [82] and [83], hold.

2. If C is nonsingular, then

$$\Delta_{B,A}(C^{-1}) = BC^{-1}\Delta_{A,B}(C)B^{-1}C^{-1}$$

if B is nonsingular and

$$\Delta_{B,A}(C^{-1}) = C^{-1}A^{-1}\Delta_{A,B}(C)C^{-1}A$$

if A is nonsingular (see [83]).

3. $\text{rank}(\Delta_{A,B}(C)) = \text{rank}(\Delta_{B,A}(C^{-1}))$ if C is nonsingular (see [63], Theorem 1).

The connection between the displacement of a matrix and the displacement of its inverse is more complex in the case of Stein operators, as it relies on the nonsingularity of one of the matrices A and B . However, there is appeal in the ease with which one can reconstruct a matrix from its Stein displacement.

The objective with displacement is to compress the information that parametrizes a matrix into a low-rank representation. For a matrix $C \in \mathbb{C}^{m \times n}$, if the displacement of the matrix can be written as $\nabla_{A,B}(C) = GH^T$ (or $\Delta_{A,B}(C) = GH^T$) with $G \in \mathbb{C}^{m \times r}$ and $H \in \mathbb{C}^{n \times r}$, the matrices G and H are referred to as the **generator matrices** of C . The columns of G and H are referred to as the **generators** of C .

1.4.3 Singularity and recoverability

A displacement operator is said to be nonsingular when $\nabla_{A,B}(C) = \mathbf{0}$ implies $C = \mathbf{0}$ (and similarly for $\Delta_{A,B}(C)$). One condition for the nonsingularity of displacement operators is as follows.

Proposition 1.1. *The operator $\nabla_{A,B}(\cdot)$ is nonsingular if and only if $\lambda_i(A) \neq \lambda_j(B)$ for all pairs of eigenvalues $(\lambda_i(A), \lambda_j(B))$; the operator $\Delta_{A,B}(\cdot)$ is nonsingular if and only if $\lambda_i(A)\lambda_j(B) \neq 1$ for all pairs $(\lambda_i(A), \lambda_j(B))$.*

Proof. See [85], Theorem 3.3. □

A corollary provides a slightly simpler result for Sylvester displacement.

Corollary 1.1. *If the operator $\nabla_{A,B}(\cdot)$ is nonsingular, then at least one of the matrices A or B is nonsingular.*

Proof. See [85], Corollary 3.4. □

The converse of Corollary 1.1 does not hold, however. As a counterexample, if $A = B = I_n$, the displacement of any matrix would be $\mathbf{0}$, and therefore the operator $\nabla_{A,B}(\cdot)$ is singular.

A nonsingular Sylvester displacement can be expressed in terms of a Stein displacement.

Proposition 1.2. *Let an operator $\nabla_{A,B}(\cdot)$ be nonsingular; then $A^{-1}\nabla_{A,B}(\cdot) = \Delta_{A^{-1},B}(\cdot)$ if A is nonsingular and/or $\nabla_{A,B}(\cdot) = -\Delta_{A,B^{-1}}(\cdot)B$ if B is nonsingular.*

Proof. If $\nabla_{A,B}(\cdot)$ is nonsingular, by Corollary 1.1 at least one of A and B is nonsingular. If A is nonsingular, for any properly-sized matrix C ,

$$A^{-1}\nabla_{A,B}(C) = A^{-1}(AC - CB) = C - A^{-1}CB = \Delta_{A^{-1},B}(C).$$

If B is nonsingular,

$$\nabla_{A,B}(C)B^{-1} = (AC - CB)B^{-1} = ACB^{-1} - C = -\Delta_{A,B^{-1}}(C).$$

□

Since a matrix can be fully recovered from a nonsingular displacement, the representation of the matrix under the displacement operator contains all of the information that parametrizes the matrix. In other words, the generators of a matrix with respect to a nonsingular displacement completely characterize the matrix. This concept is at the heart of the displacement-rank approach, which seeks to reduce the number of calculations necessary to perform operations with or on structured matrices by working with a compact representation of them.

1.4.4 Recovery formulas

If a displacement operator is nonsingular, a matrix can be recovered from its displacement. Proposition 1.2 allows all reconstruction results to be placed in terms of Stein operators (which are easier to manipulate for this purpose). Recovery formulas for certain types of displacement matrices A and B can be derived once a recursion from [83] is established.

Lemma 1.1. *For a matrix $C \in \mathbb{C}^{m \times n}$ and matrices $A \in \mathbb{C}^{m \times m}$, $B \in \mathbb{C}^{n \times n}$, for all $k \geq 1$*

$$C = A^k C B^k + \sum_{i=0}^{k-1} A^i \Delta_{A,B}(C) B^i. \quad (9)$$

Proof. By definition, $C = \Delta_{A,B}(C) + ACB$; iterating this identity yields

$$\begin{aligned} C &= \Delta_{A,B}(C) + ACB = \Delta_{A,B}(C) + A(\Delta_{A,B}(C) + ACB)B \\ &= A^2 C B^2 + (\Delta_{A,B}(C) + A\Delta_{A,B}(C)B) = \dots = A^k C B^k + \sum_{i=0}^{k-1} A^i \Delta_{A,B}(C) B^i. \end{aligned}$$

□

Lemma 1.1 leads to the following reconstruction theorem.

Theorem 1.1. *For a matrix $C \in \mathbb{C}^{m \times n}$ and matrices $A \in \mathbb{C}^{m \times m}$, $B \in \mathbb{C}^{n \times n}$,*

1. *if $A^q = bI_m$ (i.e., if A is b -potent of order q), then*

$$C = \left(\sum_{i=0}^{q-1} A^i \Delta_{A,B}(C) B^i \right) (I_n - bB^q)^{-1}; \quad (10)$$

2. *if $B^q = cI_n$ (i.e., if B is c -potent of order q), then*

$$C = (I_m - cA^q)^{-1} \left(\sum_{i=0}^{q-1} A^i \Delta_{A,B}(C) B^i \right); \quad (11)$$

3. *and if A or B is nilpotent of order q , then*

$$C = \sum_{i=0}^{q-1} A^i \Delta_{A,B}(C) B^i \quad (12)$$

Proof. From Lemma 1.1,

$$C - A^q C B^q = \sum_{i=0}^{q-1} A^i \Delta_{A,B}(C) B^i.$$

The three points correspond to:

$$1. C(I_n - bB^q) = \sum_{i=0}^{q-1} A^i \Delta_{A,B}(C) B^i.$$

$$2. (I_m - cA^q)C = \sum_{i=0}^{q-1} A^i \Delta_{A,B}(C) B^i.$$

$$3. C = \sum_{i=0}^{q-1} A^i \Delta_{A,B}(C) B^i.$$

□

1.4.5 Displacement operator matrices

Since displacement is framed in terms of linear operators, it is instructive to consider the operator matrices that correspond to Stein and Sylvester displacements. For a matrix $A \in \mathbb{C}^{m \times n}$, let $\mathbf{vec}(A) \in \mathbb{C}^{mn \times 1}$ be a “vectorized” version of the matrix that is constructed by appending the columns of the matrix one after another; in other words, $[\mathbf{vec}(A)]_{i+(j-1)m} = A_{i,j}$. Using this notation, the following proposition describes the operator matrices of displacement operators.

Proposition 1.3. *For matrices $C \in \mathbb{C}^{m \times n}$, $A \in \mathbb{C}^{m \times m}$, and $B \in \mathbb{C}^{n \times n}$,*

$$\mathbf{vec}(\nabla_{A,B}(C)) = \Pi_{A,B}(\mathbf{vec}(C)) \quad \text{and} \quad \mathbf{vec}(\Delta_{A,B}(C)) = \Theta_{A,B}(\mathbf{vec}(C)),$$

with $\Pi_{A,B}$ and $\Theta_{A,B}$ the $mn \times mn$ matrices

$$\Pi_{A,B} = I_n \otimes A - B^T \otimes I_m \quad \text{and} \quad (13)$$

$$\Theta_{A,B} = I_{mn} - B^T \otimes A, \quad (14)$$

where \otimes denotes the Kronecker product.

Proof. See Appendix A. □

Since the operator matrix characterizes the actions of the operator, a displacement operator is nonsingular if and only if its operator matrix is nonsingular. Given a nonsingular displacement operator, a matrix can be recovered from its displacement by applying the inverse operator matrix to the vectorized displacement (and then “matricizing” the result).

The Sylvester displacement of a matrix inverse may be obtained through the displacement of the original matrix:

$$-\nabla_{B,A}(C^{-1}) = C^{-1}\nabla_{A,B}(C)C^{-1}$$

In addition, from the proof of Proposition 1.3, the following identity holds:

$$\mathbf{vec}(ACB) = (B^T \otimes A)\mathbf{vec}(C).$$

Combining these two results, the following two identities are evident:

$$\begin{aligned} -\Pi_{B,A}\mathbf{vec}(C^{-1}) &= (C^{-T} \otimes C^{-1})\Pi_{A,B}\mathbf{vec}(C) \quad \text{and} \\ -\Pi_{A,B}\mathbf{vec}(C) &= (C^T \otimes C)\Pi_{B,A}\mathbf{vec}(C^{-1}). \end{aligned}$$

1.5 Multi-level linear algebra

As computational power and sensing sophistication have improved, the scope of signal-processing problems facing engineers has broadened considerably. Many new problems that are under consideration involve multi-dimensional signals. When multi-dimensional problems are structured, the system matrices that describe their linear-algebraic formulations exhibit structure in multiple ways, and are termed *multi-level structured matrices*. To elegantly present mathematical results for these problems, it is useful to define notions of multi-level linear algebra.

1.5.1 Basic definitions

To generate readable expressions for multi-level matrices, one must first define notions of multi-level structure and the operations that manipulate it. A matrix $A \in \mathbb{R}^{mn \times mn}$ is written

as an (m, n) **two-level matrix** when it is expressed as an $n \times n$ block matrix $A = [A_{i,j}]$ with each block $A_{i,j}$ an $m \times m$ matrix. If the matrix A is written in full as $A = [a_{r,s}]$ for $1 \leq r, s \leq mn$, then for $1 \leq i, j \leq n$ the blocks $A_{i,j}$ have coefficients $[A_{i,j}]_{k,\ell} = a_{(i-1)m+k, (j-1)m+\ell}$ for $1 \leq k, \ell \leq m$.

In this notation, a multi-level structure has only been defined for square two-level matrices, but it is also useful to define “block rows” and “block columns.” A matrix $X \in \mathbb{R}^{m \times mn}$ can be written as an (m, n) block row vector $X = [X_1 \ \dots \ X_n]$ with components $X_i \in \mathbb{R}^{m \times m}$. Similarly, a matrix $X \in \mathbb{R}^{mn \times m}$ can be expressed as an (m, n) block column vector $X = [X_1^T \ \dots \ X_n^T]^T$.

One can immediately notice from the latter definition that notation can become cumbersome. By transposing a block column vector X , one obtains a block row vector. However, the blocks of this block row vector are the *transposes* of the blocks of X , which may not be what is desired. To avoid confusion and complex notation, several matrix operations may be defined for multi-level matrices. The most obvious and straightforward are adaptations of transposition that operate on only a single structural level.

Definition 1.8 (Block-level transpose). *For $A \in \mathbb{R}^{m_1 n_1 \times m_2 n_2}$, written as an $n_1 \times n_2$ block matrix $A = [A_{i,j}]$ with blocks $A_{i,j} \in \mathbb{R}^{m_1 \times m_2}$, the **block-level transpose**, denoted A^T , is the matrix*

$$A^T = \begin{bmatrix} A_{1,1} & \cdots & A_{n_1,1} \\ \vdots & \ddots & \vdots \\ A_{1,n_2} & \cdots & A_{n_1,n_2} \end{bmatrix} \in \mathbb{R}^{m_1 n_2 \times m_2 n_1}.$$

Definition 1.9 (Blockwise transpose). *For $A \in \mathbb{R}^{m_1 n_1 \times m_2 n_2}$, written as an $n_1 \times n_2$ block matrix $A = [A_{i,j}]$ with blocks $A_{i,j} \in \mathbb{R}^{m_1 \times m_2}$, the **blockwise transpose**, denoted $A^{\mathcal{T}}$, is the matrix*

$$A^{\mathcal{T}} = \begin{bmatrix} A_{1,1}^T & \cdots & A_{1,n_2}^T \\ \vdots & \ddots & \vdots \\ A_{n_1,1}^T & \cdots & A_{n_1,n_2}^T \end{bmatrix} \in \mathbb{R}^{m_2 n_1 \times m_1 n_2}.$$

Using Definitions 1.8 and 1.9, the following fact is true.

Fact 1.3. *Given a matrix A with a defined two-level structure, $A^T = (A^T)^{\mathcal{T}} = (A^{\mathcal{T}})^{\mathcal{T}}$.*

It must be noted at this point that there is no reason why the coefficients of A must be real; similar terminology for conjugate transposes may be defined for complex matrices. However, this is a needless complication for the sake of developing theory, and the theoretical results that use these conventions hold for complex matrices if the transposition operations are replaced with conjugate transposition operations. In short, the extension of these concepts to complex matrices is both immediate and obvious.

Unlike the standard transpose, both the block-level and blockwise transposes must be explicitly defined relative to the multi-level structure of the matrix. To emphasize this point, the block-level and blockwise transpose for a $(2, 3)$ two-level matrix are not the same operations as for a $(3, 2)$ two-level matrix. To keep things simple in terms of notation, these distinctions are omitted when describing the operations, and the definition of the multi-level matrix structure for any given result is assumed to be known and contextually established.

It is possible to permute (m, n) two-level matrices to become (n, m) two-level matrices with the use of the level-swapping matrix.

Definition 1.10 (level-swapping matrix). *The (m, n) level-swapping matrix is the matrix $P \in \mathbb{R}^{mn \times mn}$ defined by*

$$P = \sum_{i=1}^n \sum_{k=1}^m e_{(i-1)m+k} \cdot e_{(k-1)n+i}^T. \quad (15)$$

From its definition, the matrix P^T is the (n, m) level-swapping matrix.

The following proposition shows how the level-swapping matrix is used to exchange the two-level structure.

Proposition 1.4. *Let A be an (m, n) two-level matrix with blocks $A_{i,j}$ having coefficients $[A_{i,j}]_{k,\ell} = a_{(i-1)m+k,(j-1)m+\ell}$ for $1 \leq i, j \leq n$ and $1 \leq k, \ell \leq m$, and let P be the (m, n) level-swapping matrix. Then $A' = P^T A P$ is an (n, m) two-level matrix with blocks $A'_{k,\ell}$ having coefficients $[A'_{k,\ell}]_{i,j} = a_{(i-1)m+k,(j-1)m+\ell}$.*

Proof. The $(i, j)^{th}$ coefficient of the $(k, \ell)^{th}$ block of A' is

$$[A'_{k,\ell}]_{i,j} = A'_{(k-1)n+i,(\ell-1)n+j} = e_{(k-1)n+i}^T A' e_{(\ell-1)n+j}.$$

From the definition of P , it follows that

$$P e_{(\ell-1)n+j} = \sum_{j'=1}^n \sum_{\ell'=1}^m e_{(j'-1)m+\ell'} \cdot e_{(\ell'-1)n+j'}^T e_{(\ell-1)n+j} = e_{(j-1)m+\ell}.$$

Similarly, $e_{(k-1)n+i}^T P^T = e_{(i-1)m+k}^T$, and therefore

$$[A'_{k,\ell}]_{i,j} = e_{(i-1)m+k}^T A e_{(j-1)m+\ell} = a_{(i-1)m+k,(j-1)m+\ell}.$$

□

1.5.2 Symmetry and persymmetry

A square matrix A is symmetric if its coefficients are reflected about its main diagonal; that is, if $A^T = A$. Similar notions of symmetry may be established among the various levels of a multi-level matrix. An (m, n) two-level matrix A is **block-level symmetric** if $A^T = A$ and **blockwise symmetric** if $A^{\mathcal{F}} = A$.

The relations between symmetries in various levels of a matrix are nuanced. If a matrix is both block-level symmetric and blockwise symmetric, it is necessarily symmetric. However, the converse does not hold; for example, while the $(m, 3)$ two-level matrix

$$A = \begin{bmatrix} A_1 & A_2 & A_3 \\ A_2^T & A_4 & A_5 \\ A_3^T & A_5^T & A_6 \end{bmatrix}$$

is symmetric when A_1 , A_4 , and A_6 are symmetric, it is only blockwise and block-level symmetric if A_2 , A_3 , and A_5 are all symmetric. The one-way nature of the relationship of symmetries is summarized with the following fact.

Fact 1.4. *Let \mathcal{S}_T , $\mathcal{S}_{\mathcal{T}}$, and $\mathcal{S}_{\mathcal{F}}$ be the sets of all symmetric, block-level symmetric, and blockwise symmetric (m, n) two-level matrices, respectively. Then:*

1. $\mathcal{S}_{\mathcal{T}} \not\subset \mathcal{S}_{\mathcal{T}}$; for example, $A = I_2 \otimes B$, where \otimes denotes the Kronecker product and B is non-symmetric, satisfies $A \in \mathcal{S}_{\mathcal{T}}$ and $A \notin \mathcal{S}_{\mathcal{T}}$.
2. $\mathcal{S}_{\mathcal{D}} \not\subset \mathcal{S}_{\mathcal{T}}$; the similar example $A = B \otimes I_2$ with B non-symmetric demonstrates this point.
3. $(\mathcal{S}_{\mathcal{T}} \cap \mathcal{S}_{\mathcal{D}}) \subset \mathcal{S}_{\mathcal{T}}$, which follows directly from Fact 1.3.

Other types of symmetries than reflection about the diagonal also exist. For example, the following is a symmetry of particular interest for Toeplitz matrices.

Definition 1.11 (Persymmetry). Define the $n \times n$ **exchange matrix** $J_n = \sum_i e_i e_{n+1-i}^T$, with e_i the i^{th} canonical vector of \mathbb{R}^n . An $n \times n$ matrix A is **persymmetric** if $J_n A^T J_n = A$.

Much as symmetry is mathematically defined relative to the transpose operation, persymmetry can be defined relative to the persymmetric transpose.

Definition 1.12 (Persymmetric transpose). The **persymmetric transpose** of a matrix $A \in \mathbb{R}^{m \times n}$ is the $n \times m$ matrix $A^P = J_n A^T J_m$.

Using the persymmetric transpose, an equivalent definition of persymmetry for a matrix A is that $A^P = A$. As is the case for symmetry, a matrix can only exhibit persymmetry if it is square. Several other properties of persymmetry are also analogs of the properties of symmetry.

Proposition 1.5. For $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$, the following hold:

1. Let $m = n$; $A^P = A$ if and only if $J_n A = A^T J_n$ and $A J_n = J_n A^T$.
2. $(A^T)^P = (A^P)^T$.
3. If $m = n$ and A is nonsingular, $(A^{-1})^P = (A^P)^{-1} = A^{-P}$.
4. $(AB)^P = B^P A^P$.

Proof. See Appendix A. □

In addition to these properties, there are the following facts on persymmetry.

Fact 1.5. *The following statements on persymmetry hold:*

1. *The sum of two persymmetric matrices is persymmetric, which follows from the linearity of transposition (and therefore of persymmetric transposition).*
2. *Powers of persymmetric matrices are also persymmetric, which follows from the fourth point of Proposition 1.5.*
3. *Square Toeplitz matrices are persymmetric since they have constant diagonals (and therefore must be symmetric about their antidiagonals).*
4. *The inverse of a nonsingular persymmetric matrix is also persymmetric from the third point of Proposition 1.5.*
5. *The inverse of a nonsingular Toeplitz matrix is necessarily persymmetric from the previous two points.*

The first statement of Fact 1.5 implies that the set of all persymmetric matrices is closed under addition. It is not, however, closed under multiplication.

Proposition 1.6. *Let $A, B \in \mathbb{R}^{n \times n}$ with $A^P = A$ and $B^P = B$. Then $AB = (AB)^P$ if and only if $AB = BA$.*

Proof. The persymmetric transpose of (AB) is given by $(AB)^P = B^P A^P = BA$. Therefore, $AB = (AB)^P$ if and only if $AB = BA$. □

The next proposition further explores the connection between persymmetry and matrix products.

Proposition 1.7. *Let $A, B \in \mathbb{R}^{n \times n}$, with A nonsingular. Then the following statements hold:*

1. $AB^P = BA^P$ if and only if $B = AC$ for $C \in \mathbb{R}^{n \times n}$ and $C^P = C$.

2. $A^P B = B^P A$ if and only if $B = CA$ for $C \in \mathbb{R}^{n \times n}$ and $C^P = C$.

Proof. See Appendix A. □

The definitions of block-level and blockwise transposes from Section 1.5 may be generalized for the persymmetric transpose as well, allowing for notions of block-level and blockwise persymmetry.

Definition 1.13 (Block-level persymmetric transpose). *Define the (m, n) block-exchange matrix $\mathcal{J}_{m,n} = J_n \otimes I_m$. Given a matrix $A \in \mathbb{R}^{m_1 n_1 \times m_2 n_2}$ written as an $n_1 \times n_2$ block matrix $A = [A_{i,j}]$ with blocks $A_{i,j} \in \mathbb{R}^{m_1 \times m_2}$, the **block-level persymmetric transpose** of A is given by*

$$A^P = \mathcal{J}_{m_1, n_2} A^T \mathcal{J}_{m_2, n_1} = \begin{bmatrix} A_{n_1, n_2} & \cdots & A_{1, n_2} \\ \vdots & \ddots & \vdots \\ A_{n_1, 1} & \cdots & A_{1, 1} \end{bmatrix}.$$

Definition 1.14 (Blockwise persymmetric transpose). *Let the (m, n) blockwise exchange matrix be denoted by $\mathcal{J}_{m,n} = I_n \otimes J_m$. Given a matrix $A \in \mathbb{R}^{m_1 n_1 \times m_2 n_2}$ written as an $n_1 \times n_2$ block matrix $A = [A_{i,j}]$ with blocks $A_{i,j} \in \mathbb{R}^{m_1 \times m_2}$, the **blockwise persymmetric transpose** of A is given by*

$$A^{\mathcal{P}} = \mathcal{J}_{m_2, n_1} A^{\mathcal{T}} \mathcal{J}_{m_1, n_2} = \begin{bmatrix} A_{1,1}^P & \cdots & A_{1, n_2}^P \\ \vdots & \ddots & \vdots \\ A_{n_1, 1}^P & \cdots & A_{n_1, n_2}^P \end{bmatrix}.$$

The following fact relates the various persymmetric transpositions to one another, and is an analog of Fact 1.3.

Fact 1.6. *For an (m, n) two-level matrix A , $A^P = (A^{\mathcal{P}})^P = (A^P)^{\mathcal{P}}$, which follows from Fact 1.3 and the commutativity of the matrices $(I_n \otimes Q)$ and $(R \otimes I_m)$ for any $Q \in \mathbb{R}^{m \times m}$, $R \in \mathbb{R}^{n \times n}$.*

An (m, n) two-level matrix A is **block-level persymmetric** if $A^P = A$ and **blockwise persymmetric** if $A^\mathcal{P} = A$. A persymmetric analog of Fact 1.4 is given as follows.

Fact 1.7. Let \mathcal{S}_P , $\mathcal{S}_\mathcal{P}$, and $\mathcal{S}_\mathcal{P}$ be the sets of all persymmetric, block-level persymmetric, and blockwise persymmetric (m, n) two-level matrices, respectively. Then:

1. $\mathcal{S}_\mathcal{P} \not\subset \mathcal{S}_P$.
2. $\mathcal{S}_\mathcal{P} \not\subset \mathcal{S}_P$.
3. $(\mathcal{S}_\mathcal{P} \cap \mathcal{S}_\mathcal{P}) \subset \mathcal{S}_P$.

The persymmetric transpose of a block column vector can be equivalently expressed with the following proposition, which proves useful when decomposing the displacement expressions of Section 1.4 for two-level Toeplitz matrices

Proposition 1.8. Let $Q \in \mathbb{R}^{mn \times m}$ be written as the block column vector $Q = [Q_i]$. Denote by \tilde{Q} the block reversal; that is, $\tilde{Q} = \mathcal{J}_{m,n}Q$. Then

$$Q^P = (\tilde{Q}^\mathcal{P})^T. \quad (16)$$

Proof. The term to be block-level transposed is given by

$$(\tilde{Q})^\mathcal{P} = \begin{bmatrix} Q_n^P \\ \vdots \\ Q_1^P \end{bmatrix}.$$

This expression implies

$$\begin{aligned} (\tilde{Q}^\mathcal{P})^T &= \begin{bmatrix} Q_n^P & \cdots & Q_1^P \end{bmatrix} = J_m \begin{bmatrix} Q_n^T & \cdots & Q_1^T \end{bmatrix} \mathcal{J}_{m,n} \\ &= J_m \begin{bmatrix} Q_1^T & \cdots & Q_n^T \end{bmatrix} \mathcal{J}_{m,n} \mathcal{J}_{m,n} = J_m Q^T J_{mn} = Q^P. \end{aligned}$$

□

CHAPTER II: CHARACTERIZATION OF STRUCTURED SYSTEMS

2.1 Displacement and generators of structured matrices

Using the properties of displacement established in Section 1.4, it is possible to derive inverse characterizations for structured matrices. For a given matrix structure, it is first necessary to identify which matrices yield compact displacement representations. Once the properties of the displacement operator are known, efficient algorithms may then be developed to exploit the given matrix structure.

2.1.1 Scalar Toeplitz matrices and inverses

To use displacement to compactly describe a matrix, it is necessary to know what type of matrices A and B in (7) and (8) capture the salient features of the matrix structure. For Toeplitz matrices, displacement yields low-rank representations when A and B are f -shift matrices.

Definition 2.1 (f -shift matrix). *The $n \times n$ f -shift matrix Z_f is defined as*

$$Z_f = \begin{bmatrix} 0 & \cdots & 0 & f \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{0}^T & 0 \\ \mathbf{I}_{n-1} & \mathbf{0} \end{bmatrix}.$$

The f -shift matrix can be used to define the class of f -circulant matrices, as any f -circulant matrix A can be written as

$$A = \sum_{i=0}^{n-1} Z_f^i a_i$$

for some $a_i \in \mathbb{C}$.

2.1.1.1 Generators for Toeplitz matrices and inverses

The displacements $\nabla_{Z_{f_1}, Z_{f_2}}(\cdot)$ and $\nabla_{Z_{f_1}^T, Z_{f_2}^T}(\cdot)$ of a Toeplitz matrix produce closed-form expressions with rank no larger than two. A convenient choice for Sylvester displacement is $f_1 = 0$, but similar results are simple to show for arbitrary values as well.

Proposition 2.1. *For a nonsingular $n \times n$ Toeplitz matrix $T = [a_{i-j}]$ and every $f, \alpha \in \mathbb{C}$,*

$$\nabla_{Z_0, Z_f}(T) = (h_{[\alpha]} - fTe_1)\hat{e}_1^T - e_1\hat{h}_\alpha^T = (h_{[\alpha]}^P - fTe_1)e_1^P - e_1h_{[\alpha]}^P \quad \text{and} \quad (17)$$

$$\nabla_{Z_0^T, Z_f^T}(T) = (g_{[\alpha]} - fTe_n)\hat{e}_n^T - e_n\hat{g}_\alpha^T = (g_{[\alpha]} - fTe_n)e_n^P - e_n g_{[\alpha]}^P, \quad (18)$$

where $h_{[\alpha]}$ and $g_{[\alpha]}$ are the vectors

$$h_{[\alpha]} = \begin{bmatrix} \alpha & a_{1-n} & \cdots & a_{-1} \end{bmatrix}^T \quad \text{and} \quad (19)$$

$$g_{[\alpha]} = \begin{bmatrix} a_1 & \cdots & a_{n-1} & \alpha \end{bmatrix}^T, \quad (20)$$

and $\hat{x} = J_n x$ denotes the reversal of the vector x .

Proof. The first displacement is shown by:

$$\begin{aligned} \nabla_{Z_0, Z_f}(T) &= Z_0 T - T Z_f \\ &= \begin{bmatrix} 0 & \cdots & 0 & 0 \\ a_0 & \cdots & a_{2-n} & a_{1-n} \\ \vdots & \ddots & \vdots & \vdots \\ a_{n-2} & \cdots & a_0 & a_{-1} \end{bmatrix} - \begin{bmatrix} a_{-1} & \cdots & a_{1-n} & fa_0 \\ a_0 & \cdots & a_{2-n} & fa_1 \\ \vdots & \ddots & \vdots & \vdots \\ a_{n-2} & \cdots & a_0 & fa_{n-1} \end{bmatrix} \\ &= (h_{[\alpha]} - fTe_1)\hat{e}_1^T - e_1\hat{h}_\alpha^T = (h_{[\alpha]} - fTe_1)(e_1^T J_n) - e_1(h_{[\alpha]}^T J_n) \\ &= (h_{[\alpha]} - fTe_1)e_1^P - e_1h_{[\alpha]}^P. \end{aligned}$$

The second is shown in similar fashion. □

As was shown in Fact 1.1, a particularly appealing aspect of Sylvester displacement is that the displacement of the inverse of a matrix may be obtained from the displacement of the matrix itself.

Proposition 2.2. For a nonsingular $n \times n$ Toeplitz matrix $T = [a_{i-j}]$, let $B = T^{-1}$. For any scalar $\alpha \in \mathbb{C}$, define vectors $u = Be_1$, $v_{[\alpha]} = Bh_{[\alpha]}$, $w = Be_n$, and $x_{[\alpha]} = Bg_{[\alpha]}$, with $h_{[\alpha]}$ and $g_{[\alpha]}$ as given in (19) and (20), respectively. Then

$$\nabla_{Z_0, Z_0}(B) = u\hat{v}_{[\alpha]}^T - v_{[\alpha]}\hat{u}^T = uv_{[\alpha]}^P - v_{[\alpha]}u^P \quad \text{and} \quad (21)$$

$$\nabla_{Z_0^T, Z_0^T}(B) = w\hat{x}_{[\alpha]}^T - x_{[\alpha]}\hat{w}^T = wx_{[\alpha]}^P - x_{[\alpha]}w^P. \quad (22)$$

Proof. From Fact 1.1, $\nabla_{Z_0, Z_0}(B) = -B\nabla_{Z_0, Z_0}(T)B$. Therefore, from (17) it follows that

$$\nabla_{Z_0, Z_0}(B) = (Be_1)(h_{[\alpha]}^P B) - (Bh_{[\alpha]})(e_1^P B).$$

By Fact 1.5, T is persymmetric (and therefore B is as well). Thus for any vector q such that $r = Bq$, it is also true that $q^P B = r^P$, and therefore

$$\nabla_{Z_0, Z_0}(B) = uv_{[\alpha]}^P - v_{[\alpha]}u^P = u\hat{v}_{[\alpha]}^T - v_{[\alpha]}\hat{u}^T.$$

Similarly, $\nabla_{Z_0^T, Z_0^T}(B) = -B\nabla_{Z_0^T, Z_0^T}(T)B$, and from (18) it is clear that

$$\begin{aligned} \nabla_{Z_0, Z_0}(B) &= (Be_n)(g_{[\alpha]}^P B) - (Bg_{[\alpha]})(e_n^P B) \\ &= wx_{[\alpha]}^P - x_{[\alpha]}w^P = w\hat{x}_{[\alpha]}^T - x_{[\alpha]}\hat{w}^T. \end{aligned}$$

□

It is also instructive to consider the Stein displacement of Toeplitz matrices.

Proposition 2.3. For an $n \times n$ Toeplitz matrix $T = [a_{i-j}]$ and every $f \in \mathbb{C}$,

$$\Delta_{Z_0, Z_f^T}(T) = e_1 h_{[0]}^P Z_0^T + (Te_1 - fh_{[0]})e_1^T \quad \text{and} \quad (23)$$

$$\Delta_{Z_0^T, Z_f}(T) = e_n g_{[0]}^P Z_0 + (Te_n - fg_{[0]})e_n^T, \quad (24)$$

where $h_{[0]}$ and $g_{[0]}$ are as given in (19) and (20), respectively.

Proof. The first displacement is shown by:

$$\begin{aligned}
\Delta_{Z_0, Z_f^T}(T) &= T - Z_0 T Z_f^T \\
&= \begin{bmatrix} a_0 & a_{-1} & \cdots & a_{1-n} \\ a_1 & a_0 & \cdots & a_{2-n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \cdots & a_0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & \cdots & 0 \\ f a_{1-n} & a_0 & \cdots & a_{2-n} \\ \vdots & \vdots & \ddots & \vdots \\ f a_{-1} & a_{n-2} & \cdots & a_0 \end{bmatrix} \\
&= e_1 h_{[0]}^P Z_0^T - f h_{[0]} e_1^T + T e_1 e_1^T,
\end{aligned}$$

and the second in a similar fashion. \square

It is also possible to describe the Stein displacement of the inverse matrix from Proposition 2.3, though it is more complex than the analogous result for Sylvester displacement.

Proposition 2.4. *For a nonsingular $n \times n$ Toeplitz matrix $T = [a_{i-j}]$, let $B = T^{-1}$. For any scalar $\alpha \in \mathbb{C}$, define vectors $u = B e_1$, $v_{[\alpha]} = B h_{[\alpha]}$, $w = B e_n$, and $x_{[\alpha]} = B g_{[\alpha]}$, where $h_{[\alpha]}$ and $g_{[\alpha]}$ are as given in (19) and (20), respectively. Then*

$$\Delta_{Z_0^T, Z_0}(B) = \dot{u} v_{[\alpha]}^P - \dot{v}_{[\alpha]} u^P \quad \text{and} \quad (25)$$

$$\Delta_{Z_0, Z_0^T}(B) = \dot{w} x_{[\alpha]}^P - \dot{x}_{[\alpha]} w^P, \quad (26)$$

where $\dot{u} = Z_0^T u$, $\dot{v}_{[\alpha]} = Z_0^T v_{[\alpha]} - e_n$, $\dot{w} = Z_0 w$, $\dot{x}_{[\alpha]} = Z_0 x_{[\alpha]} - e_1$.

Proof. See Appendix A. \square

2.1.1.2 Nullspaces of Toeplitz displacements

The operators $\nabla_{Z_0, Z_0}(\cdot)$ and $\nabla_{Z_0^T, Z_0^T}(\cdot)$ yield low-rank displacement expressions for Toeplitz matrices, but they also have the following interesting property.

Proposition 2.5. *Let $A \in \mathbb{C}^{n \times n}$; then*

1. $\nabla_{Z_0, Z_0}(A) = \mathbf{0}$ if and only if A is lower-triangular Toeplitz, and
2. $\nabla_{Z_0^T, Z_0^T}(A) = \mathbf{0}$ if and only if A is upper-triangular Toeplitz.

Proof. See Appendix A. □

The following is an obvious corollary of this result.

Corollary 2.1. *Let $A, B \in \mathbb{C}^{n \times n}$; then*

1. $\nabla_{Z_0, Z_0}(A) - \nabla_{Z_0, Z_0}(B) = \mathbf{0}$ if and only if there exists a lower-triangular Toeplitz matrix C such that $B = A + C$, and
2. $\nabla_{Z_0^T, Z_0^T}(A) - \nabla_{Z_0^T, Z_0^T}(B) = \mathbf{0}$ if and only if there exists an upper-triangular Toeplitz matrix C such that $B = A + C$.

The operators $\Delta_{Z_0, Z_0^T}(\cdot)$ and $\Delta_{Z_0^T, Z_0}(\cdot)$ were also shown to yield low-rank representations of Toeplitz matrices, but come with the added benefit that they are nonsingular.

Proposition 2.6. *Let $A \in \mathbb{R}^{n \times n}$; then*

1. $\Delta_{Z_0, Z_0^T}(A) = \mathbf{0}$ if and only if $A = \mathbf{0}$, and
2. $\Delta_{Z_0^T, Z_0}(A) = \mathbf{0}$ if and only if $A = \mathbf{0}$.

Proof. If $A = \mathbf{0}$, then certainly $\Delta_{Z_0, Z_0^T}(A) = \mathbf{0}$. Writing $A = [a_{i,j}]$, from (8),

$$\begin{aligned} \Delta_{Z_0, Z_0^T}(A) &= A - Z_0 A Z_0^T \\ &= \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} - \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & a_{1,1} & \cdots & a_{1,(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{(n-1),2} & \cdots & a_{(n-1),(n-1)} \end{bmatrix}. \end{aligned}$$

If $\Delta_{Z_0, Z_0^T}(A) = \mathbf{0}$, then $a_{i,1} = a_{1,j} = 0$ for $1 \leq i, j \leq n$ from the first row and column, and $a_{i,j} = a_{i+1,j+1}$ for $1 \leq i, j < n$ from the remaining entries. This implies $A = \mathbf{0}$.

The second point is proved in a similar fashion. □

Much as was the case for Corollary 2.1, the following corollary is an immediate consequence of Proposition 2.6.

Corollary 2.2. *Let $A, B \in \mathbb{R}^{n \times n}$; then*

1. $\Delta_{Z_0, Z_0^T}(A) - \Delta_{Z_0, Z_0^T}(B) = \mathbf{0}$ if and only if $A = B$, and
2. $\Delta_{Z_0^T, Z_0}(A) - \Delta_{Z_0^T, Z_0}(B) = \mathbf{0}$ if and only if $A = B$.

While the Sylvester displacements $\nabla_{Z_0, Z_0}(\cdot)$ and $\nabla_{Z_0^T, Z_0^T}(\cdot)$ are singular, they can be used to construct a related nonsingular Sylvester displacement.

Proposition 2.7. *The displacement operators $\nabla_{Z_1, Z_0}(\cdot)$ and $\nabla_{Z_0, Z_1}(\cdot)$ are nonsingular.*

Proof. See Appendix A. □

2.1.1.3 Toeplitz inverse formulas

It has been well established that the inverse of a Toeplitz matrix is expressible in an infinite number of Gohberg-Semencul formulas. These formulas are SSDs for Toeplitz inverses that use triangular Toeplitz components. Specifically, a Gohberg-Semencul formula decomposes a scalar Toeplitz inverse into either the form

$$T^{-1} = \sum_i U_i L_i \quad \text{or} \quad T^{-1} = \sum_i L_i U_i,$$

where the $\{L_i\}$ are lower-triangular Toeplitz matrices and the $\{U_i\}$ are upper-triangular Toeplitz matrices. In fact, all Gohberg-Semencul formulas can be written in both forms as a consequence of the persymmetry of the inverse.

Proposition 2.8. *A persymmetric matrix $A \in \mathbb{C}^{n \times n}$ can be written as $A = \sum_i Q_i R_i$, where the $\{Q_i\}$ and $\{R_i\}$ are persymmetric, if and only if it can also be written as $A = \sum_i R_i Q_i$.*

Proof. Suppose $A = \sum_i Q_i R_i$; then by the persymmetry of A

$$A = A^P = \left(\sum_i Q_i R_i \right)^P = \sum_i (Q_i R_i)^P = \sum_i R_i^P Q_i^P = \sum_i R_i Q_i.$$

A similar proof follows for the reverse direction. □

The Gohberg-Semencul formulas express the persymmetric T^{-1} as a sum of products of persymmetric matrices, and therefore each has two equivalent expressions.

While they have been derived in a number of ways, all Gohberg-Semencul formulas can be viewed as reconstructions of the matrix T^{-1} from its generators. The following lemma shows how the triangular Toeplitz matrices in the Gohberg-Semencul formulas arise from the recovery equations of Section 1.4.4.

Lemma 2.1. *For generator matrices $G, H \in \mathbb{C}^{n \times r}$,*

$$\sum_{i=0}^{n-1} (Z_0^T)^i G H^T Z_0^i = \sum_{j=1}^r U_j L_j \quad \text{and} \quad (27)$$

$$\sum_{i=0}^{n-1} Z_0^i G H^T (Z_0^T)^i = \sum_{j=1}^r \check{L}_j \check{U}_j, \quad (28)$$

where U_j is upper-triangular Toeplitz with last column $g_j = G e_j$, L_j is lower-triangular Toeplitz with last row $h_j^T = (H e_j)^T$, \check{L}_j is lower-triangular Toeplitz with last row g_j^P , and \check{U}_j is upper-triangular Toeplitz with last column $J_n h_j$.

Proof. See Appendix A. □

For the sake of generality, which will prove useful in later developments, the following theorem further relates Gohberg-Semencul formulas to displacement recovery.

Theorem 2.1. *Let $T \in \mathbb{C}^{n \times n}$ be a nonsingular Toeplitz matrix and $B = T^{-1}$ its inverse. For any vector $x \in \mathbb{C}^n$, let $L(x)$ be a lower-triangular Toeplitz matrix with last row x^T and $U(x)$ be an upper-triangular Toeplitz matrix with last column x . Let $G, H \in \mathbb{C}^{n \times r}$ be matrices with columns g_i and h_i , respectively. Then $B = \sum_{i=1}^r U(g_i) \cdot L(h_i)$ if and only if*

1. $\Delta_{Z_0^T, Z_0}(B) = G H^T$,
2. $\Delta_{Z_0, Z_0^T}(B) = J_n H G^T J_n$,
3. $\nabla_{Z_1, Z_0}(B) = Z_1 G H^T - e_1 \dot{w}^P$,
4. $\nabla_{Z_0, Z_1}(B) = \dot{w}_n^T - J_n H G^T J_n Z_1$,

$$5. \nabla_{Z_1^T, Z_0^T}(B) = Z_1^T J_n H G^T J_n - e_n \dot{u}^P, \text{ and}$$

$$6. \nabla_{Z_0^T, Z_1^T}(B) = \dot{u} e_1^T - G H^T Z_1^T.$$

Proof. See Appendix A. □

Using Theorem 2.1, displacement structure can be used to generate Gohberg-Semencul formulas and *vice versa*. Moreover, this theorem can be used to state *all* Gohberg-Semencul formulas with the following corollary.

Corollary 2.3. *Let $T \in \mathbb{C}^{n \times n}$ be a nonsingular Toeplitz matrix and $B = T^{-1}$ its inverse. For any vector $x \in \mathbb{C}^n$, let $L(x)$ and $U(x)$ be as given in Theorem 2.1. Let $G, H \in \mathbb{C}^{n \times r}$ be matrices with columns g_i and h_i , respectively. Then $B = \sum_{i=1}^r \alpha_i U(g_i) \cdot L(h_i)$ for scalars α_i if and only if*

$$GD_\alpha H^T = \begin{bmatrix} \dot{u} & \dot{v}_{[0]} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v_{[0]}^P \\ u^P \end{bmatrix},$$

where $D_\alpha = \text{diag}([\alpha_i])$.

Proof. From Theorem 2.1, $B = \sum_{i=1}^r \alpha_i U(g_i) \cdot L(h_i)$ if and only if $\Delta_{Z_0^T, Z_0}(B) = GD_\alpha H^T$. However, from Proposition 2.4,

$$\Delta_{Z_0^T, Z_0}(B) = \begin{bmatrix} \dot{u} & \dot{v}_{[0]} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v_{[0]}^P \\ u^P \end{bmatrix}.$$

This proves the corollary. □

A slightly simpler, rephrased alternative to Corollary 2.3 is stated as follows.

Corollary 2.4. *Let $T \in \mathbb{C}^{n \times n}$ be a nonsingular Toeplitz matrix and $B = T^{-1}$ its inverse. For any vector $x \in \mathbb{C}^n$, let $L(x)$ and $U(x)$ be as given in Theorem 2.1. For any nonsingular $A \in \mathbb{C}^{2 \times 2}$, define*

$$G = \begin{bmatrix} \dot{u} & \dot{v}_{[0]} \end{bmatrix} A \quad H = \left(A^P \begin{bmatrix} v_{[0]}^P \\ u^P \end{bmatrix} \right)^T.$$

Then

$$B = \frac{1}{\det(A)} (\mathbf{U}(g_1) \cdot \mathbf{L}(h_1) - \mathbf{U}(g_2) \cdot \mathbf{L}(h_2)).$$

Proof. From previous results,

$$\begin{aligned} \Delta_{Z_0^T, Z_0}(B) &= \begin{bmatrix} \dot{u} & \dot{v}_{[0]} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v_{[0]}^P \\ u^P \end{bmatrix} = \begin{bmatrix} \dot{u} & \dot{v}_{[0]} \end{bmatrix} A A^{-1} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} A^{-P} A^P \begin{bmatrix} v_{[0]}^P \\ u^P \end{bmatrix} \\ &= G A^{-1} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} A^{-P} H^T. \end{aligned}$$

Since A is nonsingular

$$\begin{aligned} A^{-1} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} A^{-P} &= \left(\frac{1}{\det(A)} \right)^2 \cdot \begin{bmatrix} a_{22} & -a_{12} \\ -a_{22} & a_{11} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} a_{11} & -a_{12} \\ -a_{21} & a_{22} \end{bmatrix} \\ &= \left(\frac{1}{\det(A)} \right)^2 \cdot \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \begin{bmatrix} a_{11} & -a_{12} \\ a_{21} & -a_{22} \end{bmatrix} = \frac{1}{\det(A)} \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \end{aligned}$$

and therefore

$$\det(A) \cdot \Delta_{Z_0^T, Z_0}(B) = G \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} H^T.$$

The formula for B then follows from Corollary 2.3. \square

Theorem 2.1 can also be used to connect the two Stein displacements of the Toeplitz inverse.

Corollary 2.5. *Let $T \in \mathbb{C}^{n \times n}$ be a nonsingular Toeplitz matrix and $B = T^{-1}$ its inverse. For matrices $G, H \in \mathbb{C}^{n \times r}$, $\Delta_{Z_0^T, Z_0}(B) = GH^T$ if and only if $\Delta_{Z_0, Z_0^T}(B) = J_n H G^T J_n$.*

Proof. Points 1 and 2 of Theorem 2.1 jointly prove the statement. \square

2.1.1.4 Equivalence of generators

Depending on the type of displacement operator used, the specific definitions of the generators of a Toeplitz inverse will vary. However, since the inverse can be recovered with

Gohberg-Semencul formulas from all of these different displacements, the inverse generators must all be equivalent in some sense. Indeed, Corollary 2.4 implies that all of the Toeplitz inverse generators belong to a 2-dimensional subspace of \mathbb{C}^n , and each displacement generates a slightly different - but linearly independent - pair of vectors from this subspace.

A consequence is that any of the “natural” generators u , $v_{[\alpha]}$, w , and $x_{[\beta]}$ that arise from using the obvious displacement operators for Toeplitz matrices can be expressed in terms of any pair of the remaining generators.

Theorem 2.2. *For a nonsingular Toeplitz matrix $T = [a_{i-j}]$, let $B = T^{-1}$. Define the generators $u = Be_1$, $v = Bh$, $w = Be_n$, and $x = Bg$, where $h = h_{[\alpha]}$ for any α and $g = g_{[\beta]}$ for any β . Similarly, define a set of vectors derived from the generators: $\dot{u} = Z_0^T u$, $\dot{v} = Z_0^T v - e_n$, $\dot{w} = Z_0 w$, and $\dot{x} = Z_0 x - e_1$. If $x_n v_1 \neq 1$ and $x_n, u_1, v_1 \neq 0$, and letting $\lambda = u_1(1 - x_n v_1)^{-1}$, the following relations hold:*

$$u = u_1 v_1^{-1} v + v_1^{-1} \dot{w} = -\lambda x_n v - \lambda \dot{x} = x_n \dot{w} - u_1 \dot{x}, \quad (29)$$

$$v = v_1 u_1^{-1} u - u_1^{-1} \dot{w} = -(\lambda x_n)^{-1} u - x_n^{-1} \dot{x} = -\lambda^{-1} \dot{w} - v_1 \dot{x}, \quad (30)$$

$$w = v_1 \dot{u} - u_1 \dot{v} = x_n^{-1} \dot{u} + u_1 x_n^{-1} x = -\lambda \dot{v} - \lambda v_1 x, \quad \text{and} \quad (31)$$

$$x = -\lambda^{-1} \dot{u} - x_n \dot{v} = -u_1^{-1} \dot{u} + x_n u_1^{-1} w = -v_1^{-1} \dot{v} - (\lambda v_1)^{-1} w. \quad (32)$$

Proof. See Appendix A. □

Theorem 2.2 and Corollary 2.3 can jointly be used to obtain a Gohberg-Semencul formula for any pair of generators in the set $\{u, v, w, x\}$.

Theorem 2.3. *For a nonsingular Toeplitz matrix $T = [a_{i-j}]$, let $B = T^{-1}$. Define the vectors $\{u, \dot{u}, v, \dot{v}, w, \dot{w}, x, \dot{x}\}$ as in Theorem 2.2, and for any vector $x \in \mathbb{C}^n$ let $L(x)$ and $U(x)$ be as given in Theorem 2.1. Suppose $x_n v_1 \neq 1$ and $x_n, u_1, v_1 \neq 0$ so that $\lambda = u_1(1 - x_n v_1)^{-1}$ is well-defined. For matrices $G, H \in \mathbb{C}^{n \times r}$ and diagonal matrices $D = \text{diag}(d) \in \mathbb{C}^{r \times r}$, let*

$\mathcal{G}(GDH^T)$ denote the Gohberg-Semencul formula

$$\mathcal{G}(GDH^T) = \sum_{i=1}^r d_i U(Ge_i) L(He_i).$$

Then the following are true:

1. If $G = \begin{bmatrix} \dot{u} & \dot{v} \end{bmatrix}$, $H = \begin{bmatrix} v^P \\ u^P \end{bmatrix}^T$, and $D = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$, then $B = \mathcal{G}(GDH^T)$;
2. If $G = \begin{bmatrix} \dot{u} & w \end{bmatrix}$, $H = \begin{bmatrix} \dot{w}^P \\ u^P \end{bmatrix}^T$, and $D = \begin{bmatrix} -u_1^{-1} & 0 \\ 0 & u_1^{-1} \end{bmatrix}$, then $B = \mathcal{G}(GDH^T)$;
3. If $G = \begin{bmatrix} \dot{u} & x \end{bmatrix}$, $H = \begin{bmatrix} \dot{x}^P \\ u^P \end{bmatrix}^T$, and $D = \begin{bmatrix} -x_n^{-1} & 0 \\ 0 & x_n^{-1} \end{bmatrix}$, then $B = \mathcal{G}(GDH^T)$;
4. If $G = \begin{bmatrix} w & \dot{v} \end{bmatrix}$, $H = \begin{bmatrix} v^P \\ \dot{w}^P \end{bmatrix}^T$, and $D = \begin{bmatrix} v_1^{-1} & 0 \\ 0 & -v_1^{-1} \end{bmatrix}$, then $B = \mathcal{G}(GDH^T)$;
5. If $G = \begin{bmatrix} x & \dot{v} \end{bmatrix}$, $H = \begin{bmatrix} v^P \\ \dot{x}^P \end{bmatrix}^T$, and $D = \begin{bmatrix} -\lambda & 0 \\ 0 & \lambda \end{bmatrix}$, then $B = \mathcal{G}(GDH^T)$; and
6. If $G = \begin{bmatrix} w & x \end{bmatrix}$, $H = \begin{bmatrix} \dot{x}^P \\ \dot{w}^P \end{bmatrix}^T$, and $D = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$, then $B = \mathcal{G}(GDH^T)$.

Proof. See Appendix A. □

2.1.2 Generalized Cauchy matrices and inverses

Unlike Toeplitz matrices, the displacement operators for generalized Cauchy matrices are chosen based on the matrix parameters. Specifically, the left displacement matrix is $D_c = \text{diag}(c)$ while the right displacement matrix is $D_d = \text{diag}(d)$, where c and d are the denominator nodes. Given these definitions, the properties of generalized Cauchy displacement are simple to study.

2.1.2.1 Generators for generalized Cauchy matrices and inverses

The displacement operators of generalized Cauchy matrices cancel out the contributions of the denominators of each matrix entry, leaving the terms forming the numerators as the generators.

Proposition 2.9. *Let a generalized Cauchy matrix R have entries $R_{i,j} = (z_i^T y_j)(c_i - d_j)^{-1}$, where z_i^T is the i^{th} row of $Z \in \mathbb{C}^{m \times r}$, y_j^T is the j^{th} row of $Y \in \mathbb{C}^{n \times r}$, and $c_i \neq d_j$ for all i, j . Let $D_c = \text{diag}(c)$ and $D_d = \text{diag}(d)$. Then $\nabla_{D_c, D_d}(R) = ZY^T$.*

Proof. Consider the entries of the displacement:

$$\begin{aligned} [\nabla_{D_c, D_d}(R)]_{i,j} &= (D_c R)_{i,j} - (R D_d)_{i,j} = c_i R_{i,j} - R_{i,j} d_j \\ &= \frac{(z_i^T y_j)(c_i - d_j)}{c_i - d_j} = z_i^T y_j = (ZY^T)_{i,j}. \end{aligned}$$

□

The inverse of a generalized Cauchy matrix has a similar displacement, but the diagonal matrices exchange positions.

Proposition 2.10. *Let a generalized Cauchy matrix R be defined as in Proposition 2.9. Suppose R is nonsingular and let $D_c = \text{diag}(c)$ and $D_d = \text{diag}(d)$. Then $\nabla_{D_d, D_c}(R^{-1}) = (-R^{-1}Z)(R^{-T}Y)^T$.*

Proof. The proof is immediate by applying point 2 of Fact 1.1. □

Since the inverse of a generalized Cauchy matrix is displaced by diagonal matrices on both sides, it is generalized Cauchy as well. Denote the generators of R^{-1} as $U = -R^{-1}Z$ and $V = R^{-T}Y$, and let their rows be u_i^T and v_j^T , respectively. Then

$$R_{i,j}^{-1} = \frac{u_i^T v_j}{d_i - c_j}.$$

Therefore, the problem of finding the inverse of a generalized Cauchy matrix exactly corresponds to determining its generators since the denominators of its entries are already known.

2.1.2.2 Nullspaces of generalized Cauchy displacements

For R to be a proper Cauchy matrix, $c_i \neq d_j$ for all i and j . This condition also implies that the eigenvalues of D_c and D_d are mutually distinct, and therefore the displacement of a true generalized Cauchy matrix is necessarily nonsingular. As a result, any matrix A whose displacement is given as

$$\nabla_{D_c, D_d}(A) = ZY^T$$

must be the generalized Cauchy matrix with coefficients $A_{i,j} = (z_i^T y_j)(c_i - d_j)^{-1}$. Therefore, a generalized Cauchy matrix can be restored from its generators by an operator that inverts the displacement:

$$\left[\widetilde{\nabla}^{-1}(Z, Y, c, d) \right]_{ij} = \frac{z_i^T y_j}{c_i - d_j}.$$

However, if the definition of a generalized Cauchy matrix is slightly relaxed, one may consider singular displacements. Specifically, suppose that $c_i = d_j$ for some subset of indices (i, j) , and define a “quasi-generalized Cauchy matrix”

$$G_{i,j} = \begin{cases} \frac{z_i^T y_j}{c_i - d_j} & c_i \neq d_j \\ \gamma_{i,j} & \text{else} \end{cases},$$

where the $\gamma_{i,j}$ are some specified values. The displacement of this matrix is then given by

$$(D_c G - G D_d)_{i,j} = \begin{cases} z_i^T y_j & c_i \neq d_j \\ 0 & \text{else.} \end{cases} \quad (33)$$

If it so happens that $z_i^T y_j = 0$ for all (i, j) such that $c_i = d_j$, then this expression reduces to $\nabla_{D_c, D_d}(G) = ZY^T$.

From (33), it is obvious that the nullspace of the displacement $\nabla_{D_c, D_d}(\cdot)$ then consists of the set of all matrices of the form

$$A_{i,j} = \begin{cases} a_{i,j} & c_i = d_j \\ 0 & \text{else} \end{cases}.$$

Correspondingly, the matrix G can be recovered from its displacement only up to its values at these points. However, since the displacement results in zeros in locations of shared denominator nodes, the information at these entries is lost during displacement.

2.1.2.3 Displacement of generalized Cauchy Gramians

The Gramians of generalized Cauchy matrices and their inverses have displacement structure similar to generalized Cauchy matrices, but their displacements are singular.

Corollary 2.6. *Let R be defined as in Proposition 2.9 and $G = R^T R$ be its Gramian. Let $V = Y$ and $U = R^T Z$; then*

$$\nabla_{D_d, D_d}(G) = UV^T - VU^T. \quad (34)$$

Moreover, if G is nonsingular, then

$$\nabla_{D_d, D_d}(G^{-1}) = PQ^T - QP^T, \quad (35)$$

where $P = G^{-1}Y$ and $Q = R^\dagger Z = G^{-1}R^T Z$.

Proof. For the displacement of G , from point 1 of Fact 1.1,

$$\begin{aligned} \nabla_{D_d, D_d}(G) &= \nabla_{D_d, D_c}(R^T)R + R^T \nabla_{D_c, D_d}(R) \\ &= -YZ^T R + R^T ZY^T = UV^T - VU^T. \end{aligned}$$

If G is nonsingular, then $P = G^{-1}Y = G^{-1}V$ and $Q = R^\dagger Z = G^{-1}R^T Z = G^{-1}U$. From point 2 of Fact 1.1,

$$\nabla_{D_d, D_d}(G^{-1}) = -G^{-1}UV^T G^{-1} + G^{-1}VU^T G^{-1} = PQ^T - QP^T.$$

□

From the results of Section 2.1.2.2, the displacement $\nabla_{D_d, D_d}(\cdot)$ is singular. As a result, the Gramian G cannot be recovered from its displacement, as

$$(D_d A - A D_d)_{i,i} = A_{i,i}(d_i - d_i) = 0.$$

The displacement is thus zero along the main diagonal; the operator $\nabla_{D_d, D_d}(\cdot)$ destroys any information about these coefficients.

Since the displacement is singular, there is no inverse displacement operator $\widetilde{\nabla}^{-1}(\cdot)$. Instead, an operator that recovers all but the main diagonal of $A_{i,j}$ can be used. Let $\nabla_{D_d, D_d}(A) = ZY^T$; then the matrix-valued operator $\nabla^\dagger(Z, Y, d)$ defined by

$$[\nabla^\dagger(Z, Y, d)]_{ij} = \begin{cases} \frac{z_i^T y_j}{d_i - d_j} & i \neq j \\ 0 & \text{else} \end{cases}, \quad (36)$$

where z_i^T and y_j^T are the rows of Z and Y , respectively, satisfies

$$\nabla^\dagger(Z, Y, d) = A - \text{diag}(A_{i,i})$$

if $d_i \neq d_j$ for all $i \neq j$.

It is evident that this operator is the pseudoinverse of the displacement. From Proposition 1.3, the displacement operator matrix of $\nabla_{D_d, D_d}(\cdot)$ is

$$\Pi_{D_d, D_d} = I_n \otimes D_d - D_d \otimes I_n.$$

The operator matrix is diagonal, and the $(i + in)^{\text{th}}$ entries of the diagonal are equal to zero.

The pseudoinverse of this matrix is then $\Pi_{D_d, D_d}^\dagger = \text{diag}(\lambda)$, where

$$\lambda_{i+jn} = \begin{cases} \frac{1}{d_i - d_j} & i \neq j \\ 0 & \text{else} \end{cases}.$$

The product $\Pi_{D_d, D_d}^\dagger \Pi_{D_d, D_d} \mathbf{vec}(A)$ then has values

$$\left(\Pi_{D_d, D_d}^\dagger \Pi_{D_d, D_d} \mathbf{vec}(A) \right)_{i+jn} = \begin{cases} \frac{d_i - d_j}{d_i - d_j} A_{ij} = A_{ij} & i \neq j \\ 0 & \text{else} \end{cases},$$

which is exactly the same result returned by $\nabla^\dagger(Z, Y, d)$.

2.2 Polynomial interpolation and structured systems

Section 2.1 characterized structured matrices by establishing formulas for their generators. Then, using Sylvester displacement, the generators of a structured matrix were connected to the generators of the inverse with simple identities. Once known, the inverse generators could be used to solve structured systems with reduced operation counts. These results are only useful, however, if efficient algorithms exist to actually calculate the inverse generators. This section details one approach to this problem, which relates the computation of inverse generators to specialized polynomial interpolation problems.

2.2.1 Generalized polynomial interpolation

In standard polynomial interpolation, the goal is to find a polynomial $p(z)$ such that $p(z_i) = y_i$, where z_i are the specified interpolation nodes and y_i are prescribed values. The process of interpolation then amounts to solving the structured linear system

$$\underbrace{\begin{bmatrix} 1 & z_1 & \cdots & z_1^{n-1} \\ 1 & z_2 & \cdots & z_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & z_m & \cdots & z_m^{n-1} \end{bmatrix}}_V \underbrace{\begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{n-1} \end{bmatrix}}_p = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}}_y.$$

From this equation one can see that the matrix V is a Vandermonde matrix parametrized by the nodes z_i , and methods of solving these systems are well studied.

Standard interpolation is encountered in a limited range of applications, but it can be generalized to more complicated problems. **Rational interpolation** aims to find polynomials $p(z)$ and $q(z)$ such that the rational function $R(z) = p(z)/q(z)$ satisfies

$$R(z_i) = \frac{p(z_i)}{q(z_i)} = y_i$$

for interpolation nodes z_i and values y_i . For these problems to be well-defined, the total number of degrees of freedom in $p(z)$ and $q(z)$ should equal the number of interpolation

conditions. When expanded into large linear systems, rational-interpolation problems are written as

$$\underbrace{\begin{bmatrix} 1 & z_1 & \cdots & z_1^{n_1-1} \\ 1 & z_2 & \cdots & z_2^{n_1-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & z_m & \cdots & z_m^{n_1-1} \end{bmatrix}}_{V_{1:n_1}} \underbrace{\begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{n_1-1} \end{bmatrix}}_p = \underbrace{\begin{bmatrix} y_1 & 0 & \cdots & 0 \\ 0 & y_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & y_m \end{bmatrix}}_{D_y} \underbrace{\begin{bmatrix} 1 & z_1 & \cdots & z_1^{n_2-1} \\ 1 & z_2 & \cdots & z_2^{n_2-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & z_m & \cdots & z_m^{n_2-1} \end{bmatrix}}_{V_{1:n_2}} \underbrace{\begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_{n_2-1} \end{bmatrix}}_p,$$

where the subscript $V_{1:k}$ denotes a submatrix formed from the first k columns of a larger matrix V . These problems may be alternatively expressed as linear homogeneous systems

$$\begin{bmatrix} V_{1:n_1} & -D_y V_{1:n_2} \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} = \mathbf{0},$$

and the matrices of these systems are referred to as **coupled Vandermonde** [66].

Tangential-interpolation problems are even more general in scope, and aim to find polynomials $\{p_j(z)\}$ such that

$$\lambda_{i,1}p_1(z_i) + \lambda_{i,2}p_2(z_i) + \cdots + \lambda_{i,r}p_r(z_i) = 0$$

for some specified scalars $\lambda_{i,j}$ and polynomials $p_j(z)$. Each interpolation condition can be written as the inner product between an interpolation condition $\lambda_i = [\lambda_{i,1} \ \cdots \ \lambda_{i,r}]^T$ and a vector polynomial $\mathbf{p}(z) = [p_1(z) \ \cdots \ p_r(z)]^T$:

$$\lambda_i^T \mathbf{p}(z_i) = \begin{bmatrix} \lambda_{i,1} & \cdots & \lambda_{i,r} \end{bmatrix} \begin{bmatrix} p_1(z_i) \\ \vdots \\ p_r(z_i) \end{bmatrix} = 0.$$

Expressing the interpolation conditions as a large linear system, tangential-interpolation problems correspond to linear homogeneous systems of the form

$$\begin{bmatrix} \Lambda_1 V_{n_1} & \cdots & \Lambda_r V_{n_r} \end{bmatrix} \begin{bmatrix} p_1 \\ \vdots \\ p_r \end{bmatrix} = \mathbf{0},$$

where $\Lambda_j = \text{diag}([\lambda_{i,j}])$, V_{n_j} is the $m \times n_j$ Vandermonde matrix parametrized by nodes z_i , and p_j is the vector of the monomial coefficients of the polynomial $p_j(z)$. From this formulation, it is evident that rational interpolation is simply a specialized form of tangential interpolation.

2.2.2 Structured systems and tangential interpolation

Using some basic manipulations, many problems of structured linear algebra can be expressed in terms of tangential interpolation. Three of particular interest are determining the inverse generators of a Toeplitz matrix, applying the pseudoinverse of a Toeplitz matrix, and solving generalized Cauchy systems. The above is far from a comprehensive list of structured problems to which tangential interpolation can be applied, but they serve as particularly illustrative examples. Indeed, Section 2.2.2.1 provides a general framework for expressing any system with Toeplitz blocks as a tangential-interpolation problem.¹

2.2.2.1 Extension and transformation for Toeplitz systems

To connect Toeplitz systems with tangential-interpolation problems, one can use a process known as “extension and transformation” (ET) first given in [52]. ET replaces Toeplitz blocks with circulant-like blocks (“extension”), which are then factored into products of diagonal and Fourier matrices (“transformation”). Since the Fourier matrix is Vandermonde, the connection to tangential interpolation is then immediate.

The first stage of the approach is to introduce circulant structure into the system by adding additional equations. The new equations correspond to additional rows, which allow the Toeplitz blocks of the original linear system to form submatrices of circulant matrices. These new blocks are referred to as **truncated circulant matrices** to avoid ambiguities in nomenclature. The end result of extension is a larger set of equations, but one with the same amount of information as the original system.

It is simplest to demonstrate extension on a single $m \times n$ Toeplitz block $T = [a_{i-j}]$ and

¹As is shown in Section 2.2.2.4, this procedure is unnecessary for generalized Cauchy systems.

system of equations $Tx = b$. Define an integer $N = m + n + k$ for any $k \geq -1$; if \tilde{T} is the $(n + k) \times n$ Toeplitz matrix

$$\tilde{T} = \begin{bmatrix} a_{-n-k} & a_{m-1} & \cdots & a_{m-n+1} \\ a_{-n-k+1} & a_{-n-k} & \cdots & a_{m-n+2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{-1} & a_{-2} & \cdots & a_{-n} \end{bmatrix},$$

where the coefficients a_i are arbitrary for $i \leq -n$, then the $N \times n$ matrix

$$C_k(T) = \begin{bmatrix} \tilde{T} \\ T \end{bmatrix} \begin{matrix} n+k \\ m \end{matrix} \quad (37)$$

consists of the first n columns of the $N \times N$ circulant matrix

$$\text{circ}_k(T) = \begin{bmatrix} a_{-n-k} & \cdots & a_{-n-k+1} \\ \vdots & \ddots & \vdots \\ a_{m-1} & \cdots & a_{-n-k} \end{bmatrix};$$

that is, $C_k(T) = (\text{circ}_k(T))_{1:n}$. The matrix \tilde{T} is the **extension matrix** of T , while $C_k(T)$ is the *circulant extension* of T . While $C_k(T)$ is not itself circulant, it is an $N \times n$ submatrix of the $N \times N$ circulant matrix $\text{circ}_k(T)$, and is therefore truncated circulant. An example is shown in Figure 1.

To replace the Toeplitz block T with the truncated circulant matrix $C_k(T)$, it is necessary to introduce additional columns to compensate for the added rows of $C_k(T)$. The extra columns are generated by defining an artificial variable $\gamma = -\tilde{T}x$. Letting \mathbf{I}_{n+k} denote the $(n + k) \times (n + k)$ identity matrix, the equations

$$\begin{matrix} & n & n+k \\ n+k & \begin{bmatrix} \tilde{T} & \mathbf{I}_{n+k} \end{bmatrix} \\ m & \begin{bmatrix} T & \mathbf{0} \end{bmatrix} \end{matrix} \begin{bmatrix} x \\ \gamma \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ b \end{bmatrix} \quad (38)$$



Figure 1: An example of circulant extension. The large matrix is circulant, while the lower-right corner T is a rectangular Toeplitz matrix. The upper-left corner, \tilde{T} , is the extension matrix. Together, T and \tilde{T} form the circulant extension $C_k(T)$.

are equivalent to the original system $Tx = b$. The introduction of the matrix \tilde{T} then imparts no additional constraints on the solution x .

The blocks of (38) can next be grouped together as truncated circulant matrices:

$$\begin{bmatrix} C_k(T) & (\mathbf{I}_N)_{1:(n+k)} \end{bmatrix} \begin{bmatrix} x \\ \gamma \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ b \end{bmatrix}.$$

The reformulation in (39) proves useful, as the truncated circulant matrices can be factored with the Fourier matrix. For example,

$$C_k(T) = \left(\mathbf{F}^H \Lambda \mathbf{F} \right)_{(1:n)} = \mathbf{F}^H \Lambda \mathbf{F}_{(1:n)}, \quad (39)$$

where $\Lambda = \sqrt{N} \cdot \text{diag}(\mathbf{F}(C_k(T)e_1))$ as in Section 1.2.1.2. Since $(\mathbf{I}_N)_{1:(n+k)}$ is truncated circulant as well, it has a similar decomposition.

This approach has an obvious generalization to larger block Toeplitz systems as well. For the 2×2 block system

$$\mathbf{T}\mathbf{x} = \begin{bmatrix} T_{1,1} & T_{1,2} \\ T_{2,1} & T_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

with $T_{i,j} \in \mathbb{C}^{m_i \times n_j}$ Toeplitz, if the extension matrices are chosen such that $\tilde{T}_{i,1}$ and $\tilde{T}_{i,2}$ have

the same number of rows then the extended system is given as

$$\mathbf{C} \tilde{\mathbf{x}} = \begin{bmatrix} \tilde{T}_{1,1} & \tilde{T}_{1,2} & \mathbf{I}_{N_1-m_1} & \mathbf{0} \\ T_{1,1} & T_{1,2} & \mathbf{0} & \mathbf{0} \\ \tilde{T}_{2,1} & \tilde{T}_{2,2} & \mathbf{0} & \mathbf{I}_{N_2-m_2} \\ T_{2,1} & T_{2,2} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \gamma_1 \\ \gamma_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ b_1 \\ \mathbf{0} \\ b_2 \end{bmatrix}.$$

After grouping them together, each set of 2×1 blocks forms a truncated circulant matrix that can be factored with a Fourier matrix. Further generalization to a $c \times d$ block pattern of Toeplitz matrices follows immediately.

Systems composed entirely of truncated circulant matrices may be transformed with Fourier operators into tangential-interpolation problems. Let an $\mathbf{N} \times \mathbf{N}$ nonsingular system $\mathbf{C}\mathbf{x} = \mathbf{y}$ be written as the $c \times d$ block system

$$\begin{bmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ C_{c,1} & C_{c,2} & \cdots & C_{c,d} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_c \end{bmatrix}, \quad (40)$$

where the blocks $C_{i,j} \in \mathbb{C}^{N_i \times n_j}$ are each tall, truncated circulant matrices (and thus $c \leq d$). This formulation also implies that $\sum_i N_i = \sum_j n_j = \mathbf{N}$. Without loss of generality, it is simplest to assume $N_i = N$ for all i (so that $\mathbf{N} = cN$), but all theoretical developments hold for more general cases as well.

Since each $C_{i,j}$ is truncated circulant, it can be diagonalized as $C_{i,j} = \mathbf{F}^H \Lambda_{i,j} \mathbf{F}_{1:n_j}$ with the entries of the diagonal matrix $\Lambda_{i,j}$ determined by a Fourier transform of the first column of $C_{i,j}$. By left-multiplying (40) with $\mathbf{I}_c \otimes \mathbf{F}$ and re-arranging the equations to form a homogenous system, the original system is reduced to

$$\begin{bmatrix} \Lambda_{11} \mathbf{F}_{1:n_1} & \cdots & \Lambda_{1d} \mathbf{F}_{1:n_d} & D_{\hat{y}_1} \mathbf{F}_{1:1} \\ \vdots & \ddots & \vdots & \vdots \\ \Lambda_{c1} \mathbf{F}_{1:n_1} & \cdots & \Lambda_{cd} \mathbf{F}_{1:n_d} & D_{\hat{y}_c} \mathbf{F}_{1:1} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_d \\ -1 \end{bmatrix} = \mathbf{0}, \quad (41)$$

where $D_{\hat{y}_i} = \text{diag}(\mathbf{F}y_i)$. Each Fourier submatrix $\mathbf{F}_{1:k}$ is Vandermonde, and thus after ET the system corresponds to a tangential-interpolation problem.

2.2.2.2 Toeplitz inverse generators

The problem of solving for the Toeplitz inverse generators through tangential interpolation was first explored in [110]. This result used the characterizing “generator function” of the inverse, as well as several clever manipulations. However, ET provides a more direct and general method of reframing the problem.

From Proposition 2.2, the generators u and $v_{[\alpha]}$ of a Toeplitz inverse satisfy

$$\begin{bmatrix} a_0 & a_{-1} & \cdots & a_{1-n} \\ a_1 & a_0 & \cdots & a_{2-n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \cdots & a_0 \end{bmatrix} \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \\ \vdots & \vdots \\ u_n & v_n \end{bmatrix} = \begin{bmatrix} 1 & \alpha \\ 0 & a_{1-n} \\ \vdots & \vdots \\ 0 & a_{-1} \end{bmatrix}.$$

This system of equations may also be written as

$$\underbrace{\begin{bmatrix} a_0 & a_{-1} & \cdots & a_{1-n} & \alpha \\ a_1 & a_0 & \cdots & a_{2-n} & a_{1-n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1} & a_{n-2} & \cdots & a_0 & a_{-1} \end{bmatrix}}_{\check{T}} \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \\ \vdots & \vdots \\ u_n & v_n \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}.$$

The matrix $\check{T} \in \mathbb{C}^{n \times (n+1)}$ is also Toeplitz, but has an additional column. After extension with n rows, the system contains truncated circulant blocks:

$$\begin{bmatrix} \check{T} & -\mathbf{I}_n \\ \check{T} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mu & \nu \\ r_u & r_v \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ e_1 & \mathbf{0} \end{bmatrix},$$

where r_u and r_v are of length n . Put into a homogeneous form, this system is

$$\begin{bmatrix} \check{T} & -\mathbf{I}_n & 0 \\ \check{T} & \mathbf{0} & -e_1 \end{bmatrix} \begin{bmatrix} \mu & \nu \\ r_u & r_v \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} \text{circ}_{-1}(\check{T}) & (-\mathbf{I}_{2n})_{1:(n+1)} \end{bmatrix} \begin{bmatrix} \mu & \nu \\ \gamma_\mu & \gamma_\nu \end{bmatrix} = \mathbf{0}.$$

While the vectors μ and γ_ν are each of length $(n+1)$, the polynomials $\mu(z)$ and $\gamma_\nu(z)$ are only of degree $(n-1)$ since their last coefficient is 0. By contrast, the polynomials $\nu(z)$ and $\gamma_\mu(z)$ are each monic polynomials of degree n . After transformation, the system is

$$\begin{bmatrix} D_{\hat{a}} \mathbf{F}_{1:(n+1)} & -\mathbf{F}_{1:(n+1)} \end{bmatrix} \begin{bmatrix} \mu & \nu \\ \gamma_\mu & \gamma_\nu \end{bmatrix} = \mathbf{0},$$

where

$$\hat{a}_k = \left(\mathbf{F} \cdot \text{circ}_{-1}(\check{T}) e_1 \right)_k = \alpha + \sum_{i=0}^{2n-1} a_{1-n+i} \omega_k^{i+1} = a_{[\alpha]}(\omega_k).$$

The generators u and $v_{[\alpha]}$ are therefore the components of the solution to a tangential-interpolation problem, or equivalently are components of the solutions of *two* related rational interpolation problems

$$\frac{\gamma_\mu(z)}{\mu(z)} = \hat{a}_k \quad \text{and} \quad \frac{\gamma_\nu(z)}{\nu(z)} = \hat{a}_k.$$

2.2.2.3 Toeplitz pseudoinversion

Expanding the work of [110], but employing the ET approach, tangential interpolation can also be used to compute the solution to overdetermined least-squares Toeplitz problems [111]. For these problem, the matrix $T \in \mathbb{C}^{m \times n}$ is tall and assumed to have full column rank. The least squares solution x^* to the system $Tx = b$ minimizes the residual $\varepsilon = (b - Tx^*)$, which is characterized by the condition $T^H \varepsilon = \mathbf{0}$. Together, these two equations form the $(m+n) \times (m+n)$ system

$$\begin{bmatrix} T & \mathbf{I}_m \\ \mathbf{0} & T^H \end{bmatrix} \begin{bmatrix} x \\ \varepsilon \end{bmatrix} = \begin{bmatrix} b \\ \mathbf{0} \end{bmatrix}.$$

Since each block of the system is Toeplitz, extension can again be applied, yielding

$$\begin{bmatrix} \widetilde{T} & \mathbf{0} & -\mathbf{I}_{n-1} & \mathbf{0} & \mathbf{0} \\ T & \mathbf{I}_m & \mathbf{0} & \mathbf{0} & -b \\ \mathbf{0} & \widetilde{T}^H & \mathbf{0} & -\mathbf{I}_{m-1} & \mathbf{0} \\ \mathbf{0} & T^H & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} x^* \\ \varepsilon \\ \gamma_{x^*} \\ \gamma_\varepsilon \\ 1 \end{bmatrix} = \mathbf{0}.$$

The transformed system is then

$$\begin{bmatrix} \Lambda_T \mathbf{F}_{1:n} & \Lambda_{1,2} \mathbf{F}_{1:m} & -\mathbf{F}_{1:(n-1)} & \mathbf{0} & \Lambda_b \mathbf{F}_{1:1} \\ \mathbf{0} & \Lambda_{T^H} & \mathbf{0} & -\mathbf{F}_{1:(m-1)} & \mathbf{0} \end{bmatrix} \begin{bmatrix} x^* \\ \varepsilon \\ \gamma_{x^*} \\ \gamma_\varepsilon \\ 1 \end{bmatrix} = \mathbf{0}.$$

Therefore, the least-squares solution x^* can be computed as the component of the solution to a tangential-interpolation problem.

2.2.2.4 Solving generalized Cauchy systems

The connection between generalized Cauchy systems and tangential interpolation is fairly direct. Let G be a nonsingular generalized Cauchy matrix with denominator nodes $c, d \in \mathbb{C}^n$ and generator matrices $U, V \in \mathbb{C}^{n \times r}$, with u_i^T and v_j^T the i^{th} and j^{th} rows of U and V , respectively. The coefficients of G are then

$$G_{i,j} = \frac{u_i^T v_j}{c_i - d_j}.$$

Define a vector rational function of the form

$$\mathbf{f}(z) = \sum_{j=1}^n v_j \cdot \frac{x_j}{z - d_j},$$

where the x_i are scalars. The function $\mathbf{f}(z)$ is vector-valued with components

$$\mathbf{f}(z) = \begin{bmatrix} f_1(z) & \cdots & f_r(z) \end{bmatrix}^T.$$

From its definition, when $z \neq d_j$, $\mathbf{f}(z)$ is well-defined and is a weighted linear combination of the column vectors v_j . This function may be used to reframe the system $Gx = b$ as an interpolation problem by noting that

$$(Gx)_i = \sum_{j=1}^n \frac{u_i^T v_j x_j}{c_i - d_j} = u_i^T \sum_{j=1}^n v_j \cdot \frac{x_j}{c_i - d_j} = u_i^T \mathbf{f}(c_i) = y_i.$$

Therefore, a vector x^* is the solution to the linear system $Gx = y$ if and only if its associated vector rational function $\mathbf{f}(z)$ satisfies the tangential interpolation conditions $u_i^T \mathbf{f}(c_i) = y_i$.

In light of these developments, generalized Cauchy systems are evidently connected with tangential-interpolation problems. However, the exact solution of these problems is considerably more nuanced than those of Toeplitz systems and involves more complicated formulations. Further details and a more thorough treatment of the use of tangential interpolation to solve generalized Cauchy systems is given in [52].

2.2.3 Solving tangential-interpolation problems

The purpose of reformulating linear-algebraic problems as tangential-interpolation problems is to solve them efficiently, which is possible because of the theoretical framework of [108]. More specifically, tangential-interpolation problems have special algebraic structure that can be exploited with a divide-and-conquer approach. This section begins with an overview (omitting many details) of the theoretical properties of tangential interpolation, and concludes by summarizing the algorithm used to solve these problems.

2.2.3.1 Theoretical foundation

For a tangential-interpolation problem whose unknown vector has d components, the solution $\mathbf{p}^*(z)$ is an element of $\mathbb{C}[z]^{d \times 1}$, the set of all $d \times 1$ vector polynomials with complex coefficients. This set is a *module*, a more abstract form of a vector space consisting of a set of elements and rules for their multiplication and addition. It is similar in nature to the vector space $\mathbb{C}^{d \times 1}$, but its elements are vectors of complex-valued polynomials rather than scalars.

The vector polynomial $\mathbf{p}^*(z)$, however, is not just any element; it also satisfies the interpolation conditions of the problem. Therefore, it is necessary to restrict the search for $\mathbf{p}^*(z)$ to the much smaller set of elements with this property. This set is defined relative to the residual of the interpolation conditions ϕ_k .

Definition 2.2 (Tangential-interpolation residual). *The **interpolation residual** of a vector polynomial $\mathbf{q}(z) \in \mathbb{C}[z]^{d \times 1}$ relative to the condition ϕ_k corresponding to the node ω_k is $(\mathcal{R}(\mathbf{q}))_k \equiv \phi_k^T \mathbf{q}(\omega_k)$.*

Using this definition, the set of vector polynomials satisfying the conditions is

$$\mathcal{S} := \left\{ \mathbf{p}(z) \in \mathbb{C}[z]^{d \times 1} : (\mathcal{R}(\mathbf{p}))_k = 0 \quad \forall k \right\}.$$

This set is a *submodule* of $\mathbb{C}[z]^{d \times 1}$, and characterizes the null space of the linear system defining the tangential-interpolation problem.

The system that defines the tangential-interpolation problem is underdetermined and homogeneous, and therefore has an infinite number of solutions. Correspondingly, the submodule \mathcal{S} is infinite, and not all of its elements are related to $\mathbf{p}^*(z)$. To illustrate, for the problems related to Toeplitz matrices the interpolation nodes are roots of unity; therefore, $(z^N - 1)\mathbf{q}(z) \in \mathcal{S}$ for any element $\mathbf{q}(z) \in \mathbb{C}[z]^{d \times 1}$. To solve the interpolation, then, one must differentiate $\mathbf{p}^*(z)$ from the irrelevant elements of \mathcal{S} .

Fortunately, the degree structure of $\mathbf{p}^*(z)$ is dictated by the extended linear system from which it is derived, and it is this information that sets it apart from the other elements of \mathcal{S} . Assuming $\mathbf{p}^*(z)$ to be the *unique* solution to the original problem, the elements of \mathcal{S} that satisfy the degree constraints can only be of the form $\eta \mathbf{p}^*(z)$, where $\eta \in \mathbb{C}$ is some scalar. A convenient tool to analyze the degree structure of the elements of \mathcal{S} is the τ -degree [108].²

Definition 2.3 (τ -degree). *Let $\tau \in \mathbb{Z}^d$ and define $\deg(0) = -\infty$. The **τ -degree** of a vector*

²In its original form, the τ -degree was referred to as the \vec{s} -degree [108]. This terminology was later changed in [110] and [111].

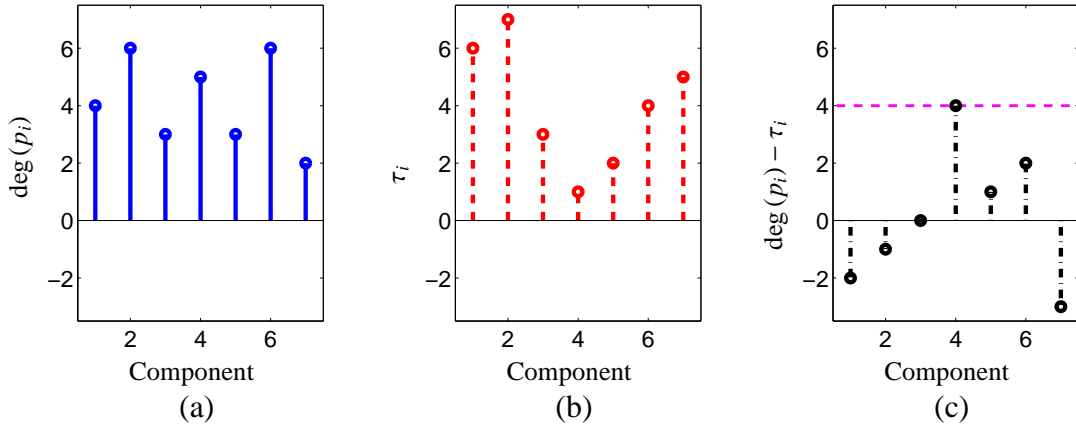


Figure 2: A visual representation of the τ -degree: (a) polynomial degrees of each component $p_i(z)$ of a 7×1 polynomial vector $\mathbf{p}(z)$; (b) individual components of a sample τ vector; (c) the degrees of $p_i(z)$ shifted by the τ values. In (c), the τ -degree is the maximum of the shifted degree values, and is represented by a dashed line.

polynomial $\mathbf{p}(z) \in \mathbb{C}[z]^{d \times 1}$ is the integer $\delta \in \mathbb{Z}$ given by

$$\delta = \tau\text{-deg}(\mathbf{p}) = \max_j (\deg(p_j) - \tau_j). \quad (42)$$

The τ -degree is the maximum degree among the components of a vector polynomial after the degree of each component has been “shifted” by a fixed amount. It is represented visually in Figure 2 for $d = 7$. For different choices of τ , the τ -degree may be different (and the components determining the τ -degree may vary as well).

As Figure 2 illustrates, the τ -degree is parametric in the sense that its definition depends on the problem context. That is, for a fixed vector polynomial $\mathbf{p}(z)$, the τ -degree depends on the supplied parameters $\{\tau_i\}$. This flexibility allows the τ -degree to reflect how closely an element of \mathcal{S} matches $\mathbf{p}^*(z)$ in degree structure for a specific problem.

To better explain, consider the τ -degree in a standard interpolation problem, where the objective is to determine the polynomial $p(z)$ of minimal degree that satisfies $p(\omega_k) = \eta_k$ for $k = 1, \dots, K$. Since there are K interpolation points, $\deg(p) = (K - 1)$. Suppose that a given polynomial $q(z)$ satisfies the conditions and that $\tau = (K - 1)$; then letting $\tau\text{-deg}(q) = k$, there are three possibilities:

- $k < 0$: $q(z)$ will not obey all conditions in general;

- $k = 0$: $q(z) = p(z)$; or
- $k > 0$: $q(z)$ is not of minimal degree.

For the vector case, letting $\tau_j = (n_j - 1)$, where $(n_j - 1)$ is the degree of each of the component polynomials of the solution, then τ -deg(\mathbf{p}^*) = 0. This value of the τ -degree then indicates that $\mathbf{p}^*(z)$ has the desired degree structure. In reality, it is not the *specific* values of $\{\tau_j\}$ that are important, but their *relative values*. For instance, if $\tilde{\tau} = \tau + \mathbf{1}$, then a $\tilde{\tau}$ -degree $\delta = -1$ indicates the proper degree structure; the shift in value merely changes how the τ -degree is interpreted for the problem.

The solution $\mathbf{p}^*(z)$ is (up to a scalar factor) the only element of \mathcal{S} that satisfies the polynomial degree constraints. Therefore, computing the solution to the original linear system amounts to determining an element of \mathcal{S} with τ -degree $\delta = 0$ for τ as given before. Once such an element is known, the actual $\mathbf{p}^*(z)$ can be computed by a re-scaling of the element to satisfy the original system.

A key property of \mathcal{S} is exploited to find such an element: it is *free*, meaning it has a basis. Submodule bases play a role similar to their linear-algebraic counterparts, allowing elements of the submodule to be described through expansions. A set $\mathbf{B}(z) = \{\mathbf{B}^{(1)}(z), \dots, \mathbf{B}^{(k)}(z)\}$ is a basis for \mathcal{S} if, for every element $\mathbf{p}(z) \in \mathcal{S}$, there are *unique* polynomials $h_i(z) \in \mathbb{C}[z]$ such that $\mathbf{p}(z)$ can be written

$$\mathbf{p}(z) = \sum_{i=1}^k h_i(z) \mathbf{B}^{(i)}(z). \quad (43)$$

The $\{h_i(z)\}$ serve as “expansion polynomials” and (43) as a basis expansion. As is the case for linear subspaces, any element of \mathcal{S} is entirely described by its $\{h_i(z)\}$ for a chosen basis, and the number of bases is infinite.

By Theorem 3.1 of [108], a basis for \mathcal{S} has exactly d elements $\mathbf{B}^{(j)}(z) \in \mathbb{C}[z]^{d \times 1}$, which may be gathered into the basis matrix polynomial $\mathbf{B}(z) = [\mathbf{B}^{(1)}(z) \ \dots \ \mathbf{B}^{(d)}(z)]$. Submodule bases play an essential role in solving tangential-interpolation problems; the solution $\mathbf{p}^*(z)$

is determined by constructing a basis for \mathcal{S} that has an element with τ -degree $\delta = 0$. Once this basis is available, the solution $\mathbf{p}^*(z)$ is then immediate.

To ensure that the basis has an element with τ -degree $\delta = 0$, one may construct a **τ -reduced** basis. A τ -reduced basis has elements that act “linearly independent” relative to the τ -degree. In other words, if δ is the maximum τ -degree among the elements of the basis, then there is no linear combination

$$\mathbf{q}(z) = \sum_i h_i(z) \mathbf{B}^{(i)}(z)$$

such that $\tau\text{-deg}(\mathbf{q}) < \delta$ other than $h_i(z) = 0$.

2.2.3.2 Basis construction

Having presented essential elements of the framework, it is now possible to give a super-fast algorithm for solving tangential-interpolation problems whose nodes are the roots of unity, beginning with a theorem that allows the construction to be subdivided into smaller problems.

Theorem 2.4. *Let $\omega_1, \dots, \omega_K$ be interpolation nodes corresponding to the conditions $\phi_1, \dots, \phi_K \in \mathbb{C}^{d \times 1}$, where the data points $\{(\omega_k, \phi_k)\}$ in the interpolation problem are mutually distinct and $\phi_k \neq \mathbf{0} \forall k$. For some $1 \leq \kappa \leq K$ and $\tau_K \in \mathbb{Z}^{d \times 1}$, let $\mathbf{B}_\kappa(z) \in \mathbb{C}[z]^{d \times d}$ be a τ_K -reduced basis corresponding to the interpolation problem $\{(\omega_k, \phi_k) : k = 1, \dots, \kappa\}$.*

Denote $\delta = [\tau_K\text{-deg}(\mathbf{B}_\kappa^{(i)})]$ for $i = 1, \dots, d$, and define $\tau_{\kappa \rightarrow K} = -\delta$. Let $\mathbf{B}_{\kappa \rightarrow K}(z) \in \mathbb{C}[z]^{d \times d}$ be a $\tau_{\kappa \rightarrow K}$ -reduced basis matrix corresponding to the interpolation problem

$$\{(\omega_k, \mathbf{B}_\kappa^T(\omega_k) \phi_k) : k = \kappa + 1, \dots, K\}.$$

Then $\mathbf{B}_K(z) = \mathbf{B}_\kappa(z) \mathbf{B}_{\kappa \rightarrow K}(z)$ is a τ_K -reduced basis matrix corresponding to the interpolation problem

$$\{(\omega_k, \phi_k) : k = 1, \dots, K\}.$$

Proof. See Van Barel and Bultheel [109], Theorem 3. □

Theorem 2.4 allows the interpolation problem to be continually subdivided into smaller subproblems. It is also the reason that a τ -reduced basis is computed, as the main result does not hold without the bases being τ -reduced.

It is still necessary to determine a solution at the finest scale, however. Suppose there is a single interpolation condition $\phi_k \in \mathbb{C}^{d \times 1}$ corresponding to the node ω_k . Without loss of generality, let τ_1 be the smallest value of τ , and define $\Delta_\ell = -(\phi_k)_\ell / (\phi_k)_1$. Then the matrix polynomial

$$\mathbf{B}(z) = \begin{bmatrix} z - \omega_k & \Delta_2 & \cdots & \Delta_d \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \quad (44)$$

satisfies $\phi_k^T \mathbf{B}(\omega_k) = \mathbf{0}^T$. Since τ_1 is the minimum τ value, $\mathbf{B}(z)$ is a τ -reduced basis for the submodule \mathcal{S}_1 defined by the single interpolation condition (ω_k, ϕ_k) (this fact is proven in [112]).

Assuming $d \ll N$, where N is the number of interpolation conditions, a basis for \mathcal{S} can be constructed in $\mathcal{O}(N^2)$ operations by processing the interpolation conditions serially with the single-point construction of (44) and invoking Theorem 2.4. This process corresponds to the “fast” basis-construction algorithms of [53] and [112]. However, the $\mathcal{O}(N^2)$ complexity can be improved by exploiting the structure of the interpolation nodes. Since the nodes are roots of unity, the number of operations can be reduced with a recursive “interleaving” splitting of the data, as given in Algorithm 1. In this routine, `TANINT` is the serial basis-construction function, which for a single point amounts to construction $\mathbf{B}(z)$ in (44). An example of how this algorithm works is given in Appendix B for a small problem.

To see how the strategy of Algorithm 1 reduces computation, the total number of required operations can be estimated. Let C_N be the total cost of calling `RECTANINT` on a set of N points. Steps ① and ③ require $C_{N/2}$ operations by definition. In step ②, the evaluations $\{\mathbf{B}_L(\omega_{2k-1})\}$ must be computed and multiplied against the $\{\phi_{2k-1}\}$. Since the

Algorithm 1 A recursive tangential-interpolation algorithm for supplied nodes $\{\omega_k\}$, conditions $\{\phi_k\}$, and τ vector.

```

procedure  $\mathbf{B}(z) = \text{RECTANINT}(\{\omega_k, \phi_k\}, \tau)$ 
   $N = \text{LENGTH}(\{\omega_k\})$ 
  if  $N = 1$  then
     $\mathbf{B}(z) \leftarrow \text{TANINT}(\{\omega_k, \phi_k\}, \tau)$ 
  else
     $\mathbf{B}_L(z) \leftarrow \text{RECTANINT}(\{\omega_{2k}, \phi_{2k}\}, \tau)$  ①
     $\delta \leftarrow \tau - \text{deg}(\mathbf{B}_L(z))$ 
    for  $k = 1, \dots, N/2$  do
       $\phi_{2k-1} \leftarrow \mathbf{B}_L(\omega_{2k-1})^T \phi_{2k-1}$  ②
    end for
     $\mathbf{B}_R(z) \leftarrow \text{RECTANINT}(\{\omega_{2k-1}, \phi_{2k-1}\}, -\delta)$  ③
     $\mathbf{B}(z) \leftarrow \mathbf{B}_L(z)\mathbf{B}_R(z)$  ④
  end if
end procedure

```

$\{\omega_k\}$ are roots of unity, $\mathbf{B}_L(z)$ can be evaluated at the nodes $\{\omega_{2k-1}\}$ using an FFT of length $N/2$. Once these values are obtained, there are a total of $d^2 N/2$ matrix-vector products. Therefore, the number of operations in step ② is upper-bounded by $\mathcal{C}_1 N \log(N)$, where \mathcal{C}_1 is a constant. Finally, step ④ requires d^2 polynomial multiplications and additions. Each multiplication can be computed with the FFT, but the FFT size depends on the degrees of the polynomials involved. By the nature of the basis construction, all of the polynomials in $\mathbf{B}_L(z)$ and $\mathbf{B}_R(z)$ must have degrees no greater than $N/2$, since they are constructed from $N/2$ conditions. Therefore, for another constant \mathcal{C}_2 , the number of operations needed to multiply $\mathbf{B}_L(z)\mathbf{B}_R(z)$ is bounded by $\mathcal{C}_2 N \log(N)$.

Adding these costs together yields the recursive formula

$$C_N = 2C_{N/2} + \mathcal{C} N \log(N),$$

where the constant \mathcal{C} is determined by the costs of steps ② and ④. Since this expression is a recurrence, the overall cost of constructing the basis can be calculated with the Master Theorem [26] to be $O(N \log^2 N)$. The key to replacing a factor of N from the fast basis construction with a factor of $\log^2(N)$ is the ability to evaluate $\mathbf{B}_L(\omega_{2k-1})$ in only $O(N \log(N))$ operations with the FFT.

The basis-construction algorithm can be unstable if not implemented carefully. Pivoting the interpolation conditions to avoid multiplying by small Δ_i values early in the construction improves the numerical stability. Without pivoting, errors may propagate as the algorithm progresses. In addition, it is possible that for a given subproblem, the remaining interpolation conditions might cause large errors in later subproblems if processed. In practice, if this is found to be the case, those conditions are skipped and included only after all remaining subproblems have been solved. The skipped conditions are marked as “difficult points,” and the calculation is more robust if they are processed at the conclusion of the construction with the fast solver. Since these points are unlikely to be evenly spaced, evaluating the basis at the difficult points requires longer FFTs. As a result, the algorithm’s efficiency worsens as the number of difficult points grows, and it is therefore important to avoid generating difficult points.

2.2.3.3 Uniqueness of tangential-interpolation solutions

Since the basis construction returns a matrix polynomial with multiple columns, an immediate question regarding the uniqueness of the solution arises. From Section 2.2.2.1, the solution to a linear-algebraic problem that has been reframed as a tangential-interpolation problem can be obtained if the τ -reduced basis for \mathcal{S} has a column with τ -degree $\delta = 0$. The solution can only truly be unique, then, if the basis has a *single* column with τ -degree $\delta_j = 0$.

Suppose that there are a total of N interpolation conditions to be processed. In each step of the algorithm, a single condition is used to increase the τ -degree of exactly one column of the current basis – that with the smallest τ -degree – by one. Initializing the basis as the identity matrix, the τ -degree of the j^{th} column is $-\tau_j$. During each iteration, the column with lowest τ -degree has its degree increased by one (while the remaining τ -degrees are unchanged). If the values of τ sum to $t = (\mathbf{N} - d + 1)$, then the sum of the final τ -degrees is $(\mathbf{N} - t) = (d - 1)$. As a result, the maximum τ -degree difference between any two columns

is one, and $(d - 1)$ of the columns will have $\delta_j = 1$ while the last has $\delta_j = 0$. Thus, a unique solution will be guaranteed.

Considering the use of tangential interpolation for Toeplitz problems, one can see that this is indeed the case for Toeplitz pseudoinversion, where

$$\tau = \begin{bmatrix} n-1 & m-1 & n-2 & m-2 & 0 \end{bmatrix},$$

and thus $t = 2(m + n - 1) - 4$. Since there are a total of $\mathbf{N} = 2(m + n - 1)$ interpolation conditions, the sum of the final τ -degrees is $(d - 1) = 4$. There are a total of five columns in the basis, so the solution is obtained by normalizing the column with τ -degree equal to 0 so that its last entry is equal to 1.

For the problem of computing the Toeplitz inverse generators, the result is obtained in a slightly different manner. The τ vector for this problem is $\tau = \begin{bmatrix} n & n \end{bmatrix}^T$, and there are a total of $2n$ interpolation conditions. Therefore, both columns of the constructed basis will have τ -degree equal to 0. However, additional information about the solutions allows them to be identified from the final basis. Specifically, since μ and γ_ν are only degree $(n - 1)$ and ν and γ_μ are monic, it follows that the matrix of their monomial coefficients for the power z^n is

$$\begin{bmatrix} \nu_n & \mu_n \\ (\gamma_\nu)_n & (\gamma_\mu)_n \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Since all of the polynomials in the constructed basis for this problem will be of degree n , the final basis can be multiplied by a 2×2 matrix on the right without changing the fact that its elements satisfy the interpolation conditions:

$$\begin{bmatrix} \hat{a}_k & -1 \end{bmatrix} \begin{bmatrix} B_{11}(\omega_k) & B_{12}(\omega_k) \\ B_{21}(\omega_k) & B_{22}(\omega_k) \end{bmatrix} \begin{bmatrix} (B_{11})_n & (B_{12})_n \\ (B_{21})_n & (B_{22})_n \end{bmatrix}^{-1} = (\phi_k B(\omega_k)) \Gamma^{-1} = \mathbf{0}^T.$$

Letting $\check{B}(z) = B(z)\Gamma^{-1}$, then

$$\begin{bmatrix} (\check{B}_{11})_n & (\check{B}_{12})_n \\ (\check{B}_{21})_n & (\check{B}_{22})_n \end{bmatrix} = \mathbf{I}_2 = \begin{bmatrix} \nu_n & \mu_n \\ (\gamma_\nu)_n & (\gamma_\mu)_n \end{bmatrix}.$$

Thus, $\nu(z) = \check{B}_{11}$ and $\mu(z) = \check{B}_{22}$, and the generators of the Toeplitz inverse can be obtained directly from these vectors.

2.3 Schur recursions

The Schur recursion is a general method of progressively obtaining the *LDU* factorization of a matrix through Schur complements.³ One condition required for the method is that the matrix to be inverted is **strongly nonsingular**, meaning all of its principal leading submatrices are nonsingular. While this condition is somewhat restrictive, the algorithm still finds use in a broad variety of applications.

The recursion functions by continually partitioning a matrix. A matrix $A \in \mathbb{C}^{2n \times 2n}$ partitioned as

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix},$$

can be factorized into a block *LDU* form as

$$A = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ A_{2,1}A_{1,1}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} A_{1,1} & \mathbf{0} \\ \mathbf{0} & S \end{bmatrix} \begin{bmatrix} \mathbf{I} & A_{1,1}^{-1}A_{1,2} \\ \mathbf{0} & \mathbf{I} \end{bmatrix},$$

where $S = A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2}$ is the Schur complement of the component $A_{1,1}$. Using this factorization, the inverse of A has the block *UDL* decomposition

$$A^{-1} = \begin{bmatrix} \mathbf{I} & -A_{1,1}^{-1}A_{1,2} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} A_{1,1}^{-1} & \mathbf{0} \\ \mathbf{0} & S^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -A_{2,1}A_{1,1}^{-1} & \mathbf{I} \end{bmatrix}.$$

As a result, to calculate A^{-1} one can subdivide the problem into four steps: calculating $A_{1,1}^{-1}$, building S from $A_{1,1}^{-1}$, calculating S^{-1} , and building the overall *UDL* decomposition from these results. For the first and third steps, the process can be recursively repeated until $A_{1,1}$ and S are scalars or sufficiently small to invert with other methods.

The computational bottleneck of the Schur recursion is the matrix multiplies that are required once the inverses are known. For generic matrices, the cost of the algorithm remains

³Though this method is used often in numerical linear algebra, use of the term ‘‘Schur recursion’’ seems to be mostly restricted to structured linear algebra.

$O(n^3)$, on par with alternative methods. For structured matrices, however, it is possible to use generators and displacement characterization to accelerate the multiplications, reducing the overall cost of the algorithm.

2.3.1 Schur recursions for Toeplitz inverses

Accelerated Schur recursions were introduced in structured linear algebra in the first superfast Toeplitz solvers [15, 77]. The original form in which the recursion was presented used a displacement-rank approach while later elaborations favored polynomial methods [3, 4]. Since the former is the more intuitive way to express the algorithms relative to the other developments thus far, it is more convenient to place the Schur recursion in the context of displacement. Moreover, the displacement-rank approach is a slightly broader way of thinking about the algorithm, as it was later extended to more general classes of structured matrices .

The keystone of the algorithm is the following theorem.

Theorem 2.5. *Let $A \in \mathbb{C}^{2n \times 2n}$ be strongly nonsingular and express it in the partitioned form*

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}.$$

Letting the size of Z_0 be implied by context, suppose $\text{rank}(A - Z_0AZ_0^T) = r$; then

$$\text{rank}(A_{1,1} - Z_0A_{1,1}Z_0^T) = \text{rank}(A_{1,1}^{-1} - Z_0^T A_{1,1}^{-1} Z_0) = r$$

and

$$\text{rank}(S - Z_0SZ_0^T) = \text{rank}(S^{-1} - Z_0^T S^{-1} Z_0) = r$$

where $S = A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2}$ is the Schur complement of $A_{1,1}$ in A .

Proof. See Appendix A. □

Theorem 2.5 uses the concept of “inheritance of displacement rank under Schur complementation.” This property is the foundation of the “complete recursive triangular factorization” (CRTF) algorithm [83], and is a feature of many superfast algorithms for structured

matrices.

Using the results of Theorem 2.5, Bitmead and Anderson broke the Schur recursion for Toeplitz inversion down into the following steps [15]:

1. Recurse to compute the generators of $A_{1,1}^{-1}$;
2. Determine the generators of $A_{1,2}$ and $A_{2,1}$ using Lemmas 2 and 6 of [15];
3. Use the computed generators to find the generators of the product $A_{2,1}A_{1,1}^{-1}A_{1,2}$;
4. Use the results of the previous step to obtain the generators of the Schur complement

$$S = A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2};$$
5. Recurse to compute the generators of $B_{2,2} = S^{-1}$;
6. Use the previous results to calculate the generators of the inverse blocks $B_{1,2} = -A_{1,1}^{-1}A_{1,2}S^{-1}$ and $B_{2,1} = -S^{-1}A_{2,1}A_{1,1}^{-1}$; and
7. Use these results to find the generators of B .

By using the FFT to perform many of the multiplications and by appealing to several Lemmas shown earlier in the work, Bitmead and Anderson (and Morf, since he used an equivalent approach) were able to show that this recursion can be calculated in $O(n \log^2 n)$ operations. These steps were generalized in the CRTF algorithm to provide superfast recursions for other classes of structured matrices as well.

2.3.2 Schur recursions for generalized Cauchy inverses

Much like Toeplitz systems, generalized Cauchy matrices may also be inverted with Schur recursion. The key principle for this version of the Schur recursion is that the displacement can be distributed among partitioned blocks of the matrix. Namely, for a nonsingular generalized Cauchy matrix R parametrized by denominator nodes $c, d \in \mathbb{C}^n$ and generator

matrices $Z, Y \in \mathbb{C}^{n \times r}$, if $D_c = \text{diag}(c)$ and $D_d = \text{diag}(d)$, the matrices may be partitioned as:

$$D_c R - R D_d = \begin{bmatrix} D_{c1} & \mathbf{0} \\ \mathbf{0} & D_{c2} \end{bmatrix} \begin{bmatrix} R_{1,1} & R_{1,2} \\ R_{2,1} & R_{2,2} \end{bmatrix} - \begin{bmatrix} R_{1,1} & R_{1,2} \\ R_{2,1} & R_{2,2} \end{bmatrix} \begin{bmatrix} D_{d1} & \mathbf{0} \\ \mathbf{0} & D_{d2} \end{bmatrix} = \begin{bmatrix} Z_1 Y_1^T & Z_1 Y_2^T \\ Z_2 Y_1^T & Z_2 Y_2^T \end{bmatrix}$$

$$\begin{bmatrix} \nabla_{D_{c1}, D_{d1}}(R_{1,1}) & \nabla_{D_{c1}, D_{d2}}(R_{1,2}) \\ \nabla_{D_{c2}, D_{d1}}(R_{2,1}) & \nabla_{D_{c2}, D_{d2}}(R_{2,2}) \end{bmatrix} = \begin{bmatrix} Z_1 Y_1^T & Z_1 Y_2^T \\ Z_2 Y_1^T & Z_2 Y_2^T \end{bmatrix}.$$

This decomposition implies that each block of the partitioned matrix has the rank- r displacement $\nabla_{D_{ci}, D_{dj}}(R_{i,j}) = Z_i Y_j^T$, and therefore each is itself a generalized Cauchy matrix.

Using a blockwise inverse, if $R_{1,1}$ is nonsingular then the inverse of R is given by

$$R^{-1} = \begin{bmatrix} R_{1,1}^{-1} + R_{1,1}^{-1} R_{1,2} S^{-1} R_{2,1} R_{1,1}^{-1} & -R_{1,1}^{-1} R_{1,2} S^{-1} \\ -S^{-1} R_{2,1} R_{1,1}^{-1} & S^{-1} \end{bmatrix},$$

where $S = R_{22} - R_{2,1} R_{1,1}^{-1} R_{1,2}$ is the Schur complement of $R_{1,1}$ in R . Consider the generator $P = -R^{-1}Z$ of R^{-1} , and partition P as R is partitioned. Then

$$\begin{aligned} -R^{-1}P &= \begin{bmatrix} R_{1,1}^{-1} + R_{1,1}^{-1} R_{1,2} S^{-1} R_{2,1} R_{1,1}^{-1} & -R_{1,1}^{-1} R_{1,2} S^{-1} \\ -S^{-1} R_{2,1} R_{1,1}^{-1} & S^{-1} \end{bmatrix} \begin{bmatrix} -Z_1 \\ -Z_2 \end{bmatrix} \\ &= \begin{bmatrix} R_{1,1}^{-1} R_{1,2} S^{-1} (Z_2 - R_{2,1} R_{1,1}^{-1} Z_1) - R_{1,1}^{-1} Z_1 \\ -S^{-1} (Z_2 - R_{2,1} R_{1,1}^{-1} Z_1) \end{bmatrix}. \end{aligned}$$

Similarly, if $Q^T = Y^T R^{-1}$, then

$$\begin{aligned} Q^T R^{-1} &= \begin{bmatrix} Y_1^T & Y_2^T \end{bmatrix} \begin{bmatrix} R_{1,1}^{-1} + R_{1,1}^{-1} R_{1,2} S^{-1} R_{2,1} R_{1,1}^{-1} & -R_{1,1}^{-1} R_{1,2} S^{-1} \\ -S^{-1} R_{2,1} R_{1,1}^{-1} & S^{-1} \end{bmatrix} \\ &= \begin{bmatrix} Y_1^T R_{1,1}^{-1} - (Y_2^T - Y_1^T R_{1,1}^{-1} R_{1,2}) S^{-1} R_{2,1} R_{1,1}^{-1} & (Y_2^T - Y_1^T R_{1,1}^{-1} R_{1,2}) S^{-1} \end{bmatrix}. \end{aligned}$$

From its inherited displacement, $R_{1,1}$ is generalized Cauchy and the displacement of its inverse is

$$\nabla_{D_{d1}, D_{c1}}(R_{1,1}^{-1}) = (-R_{1,1}^{-1} Z_i) (Y_j^T R_{1,1}^{-1}) = \widetilde{P}_1 \widetilde{Q}_1^T.$$

Since the inverse of the Schur complement S^{-1} appears in the lower right corner of R^{-1} , it follows that

$$\begin{aligned}\nabla_{D_{d_2}, D_{c_2}}(S^{-1}) &= P_2 Q_2^T = S^{-1} (R_{2,1} R_{1,1}^{-1} Z_1 - Z_2) (Y_2^T - Y_1^T R_{1,1}^{-1} R_{1,2}) S^{-1} \\ &= -S^{-1} (R_{2,1} \tilde{P}_1 + Z_2) (Y_2^T - \tilde{Q}_1^T R_{1,2}) S^{-1}.\end{aligned}$$

However, using the properties of displacement, this also implies

$$\begin{aligned}\nabla_{D_{c_2}, D_{d_2}}(S) &= -S \nabla_{D_{d_2}, D_{c_2}}(S^{-1}) S \\ &= (R_{2,1} \tilde{P}_1 + Z_2) (Y_2^T - \tilde{Q}_1^T R_{1,2}).\end{aligned}$$

Therefore, the Schur complement is also a generalized Cauchy matrix with displacement rank r . The process of computing the generators P and Q of R^{-1} involves inverting the matrices $R_{1,1}$ and S , each of which is generalized Cauchy with displacement rank r . These developments naturally lead to the Schur recursion of Algorithm 2.

When the matrix size is below some specified tolerance, the matrix R is reconstructed from its displacement and the linear systems are solved with some base $\mathcal{O}(n^3)$ algorithm. For this reason, the algorithm requires the displacement $\nabla_{D_d, D_c}(\cdot)$ to be nonsingular. If this is not the case, then at the finest level the matrix R cannot be constructed from its generators, and the solutions for the generating vectors P and Q will be incorrect.

Algorithm 2 is a reformulation of Algorithm 2.3 of [78], with the Schur complement taken on the lower right block of the matrix rather than the upper left block. It is similar in nature to the algorithm first presented by [19], but has the advantage that the generators are computed exactly and do not need to be compressed. Moreover, it requires only $\mathcal{O}(rn \log n)$ operations (see Theorem 2.4 of [78]).

The algorithm is similar to both the Gohberg-Kailath-Olshevsky (GKO) algorithm [44] and CRTF, which extends GKO to arbitrary classes of structured matrices. GKO/CRTF computes an LDU factorization of the matrix R and then solves the system with back substitution. These steps can be combined, in which case the L factor is only implicitly constructed [86].

Algorithm 2 A Schur recursion to obtain the generators of R^{-1} , the inverse of a generalized Cauchy matrix. This algorithm partitions R and computes its generators by computing the generators of the inverse of the principal leading submatrix $R_{1,1}$ and the generators of the inverse of the Schur complement S .

procedure $[P, Q] = \text{GCDISPINV}(Z, Y, c, d)$

$N \leftarrow \text{LENGTH}(Z)$

if $N < N_{min}$ **then**

$$R_{ij} \leftarrow \frac{(ZY^T)_{ij}}{c_i - d_j}$$

$$P \leftarrow -R^{-1}Z$$

$$Q \leftarrow R^{-T}Y$$

else

$$[\tilde{P}_1, \tilde{Q}_1] \leftarrow \text{GCDISPINV}(Z_1, Y_1, c_1, d_1)$$

$$Z_S \leftarrow Z_2 + R_{2,1}\tilde{P}_1$$

$$Y_S \leftarrow Y_2 - R_{1,2}^T\tilde{Q}_1$$

$$[P_2, Q_2] \leftarrow \text{GCDISPINV}(Z_S, Y_S, c_2, d_2)$$

$$P_1 \leftarrow \tilde{P}_1 - R_{1,1}^{-1}R_{1,2}P_2$$

$$Q_1 \leftarrow \tilde{Q}_1 - R_{1,1}^{-T}R_{2,1}^TQ_2$$

end if

end procedure

There are several important distinctions between GKO, CRTF, and Algorithm 2, however. First, GKO progresses serially, and thus performs $O(rn^2)$ computations for the factorization. By contrast, CRTF and Algorithm 2 both use recursive subdivision to accelerate the factorization. In CRTF, intermediate matrix multiplications are performed by exploiting the generator structure, which results in superfast algorithms. This is the exact strategy that Algorithm 2 employs; since all matrices involved are generalized Cauchy matrices in Algorithm 2, they can be applied in $O(n)$ operations with the FMM once their generators are known. Therefore, the asymptotic complexities of CRTF and Algorithm 2 are indeed superfast at $O(n \log n)$.

However, there is an important distinction between general CRTF and Algorithm 2: CRTF does not explicitly compute the inverse generators. Since both CRTF and GKO must be re-run for any new input, back substitution must also be performed again for each new

input. By contrast, Algorithm 2 computes a description of R^{-1} that may be reused for any new input. This feature allows Algorithm 2 to further reduce its operation count for new inputs.

Further computation may be saved by noting the displacement properties of the matrix products in the two final steps. The product $R_{1,1}^{-1}R_{1,2}$ has displacement

$$\begin{aligned}
\nabla_{D_{d_1}, D_{d_2}}(R_{1,1}^{-1}R_{1,2}) &= \nabla_{D_{d_1}, D_{c_1}}(R_{1,1}^{-1})R_{1,2} + R_{1,1}^{-1}\nabla_{D_{c_1}, D_{d_2}}(R_{1,2}) \\
&= \tilde{P}_1\tilde{Q}_1^T R_{1,2} + R_{1,1}^{-1}Z_1 Y_2^T \\
&= \tilde{P}_1(\tilde{Q}_1^T R_{1,2} - Y_2^T) = -\tilde{P}_1 Y_S^T.
\end{aligned}$$

Therefore, the product $R_{1,1}^{-1}R_{1,2}$ is generalized Cauchy, and its generators are known. Similarly, it can be shown that $\nabla_{D_{d_1}, D_{c_2}}(R_{1,1}^{-T}R_{2,1}^T) = -\tilde{Q}_1 Z_S^T$, and therefore the product $R_{1,1}^{-T}R_{2,1}^T$ is generalized Cauchy with known generators.

CHAPTER III: NEW ALGORITHMS FOR SCALAR STRUCTURED MATRICES

3.1 Regularized least squares for Toeplitz systems

Many real-world problems involve rectangular, singular, or poorly conditioned systems. The inversion algorithms described in Section 1.3 either are not applicable for or will fail when placed in these contexts. Expressing an archetypal example of such a system as $Tx = b$, a least-squares (LS) solution \check{x} designed to minimize the norm of the residual vector $\varepsilon = (b - T\check{x})$ is typically computed instead.

It would be beneficial in LS algorithms to again make use of Toeplitz structure to accelerate intermediate calculatory steps, and indeed there have been many adaptations of Toeplitz inversion algorithms for LS problems. As mentioned in Section 2.2.2.3 the pseudoinverse of full-column-rank systems can be computed in $O(N \log^2 N)$, where N is the number of parameters [111]. Other Toeplitz LS solvers include [55] (fast pseudoinversion), [101] (fast QR factorization), [14] and [84] (approximate, iterative calculation of generalized inverses), to name a few. This chapter extends these results by presenting a novel superfast Toeplitz Tikhonov-regularization algorithm given in [104].

3.1.1 Tikhonov regularization

Standard LS solutions are sometimes insufficient, as the vector \check{x} with the minimum residual may have undesirable properties. This difficulty arises when the pseudoinverse of the system is ill-conditioned. A more appealing solution in these instances can be computed by introducing a quadratic term that penalizes some desired signal property or properties. This general technique is known as *Tikhonov regularization*¹, and the Tikhonov-regularized

¹It is often termed “ridge regression” in the field of statistics.

solution is given as

$$x^* = \arg \min_x \|Tx - b\|^2 + \|\Gamma x\|^2. \quad (45)$$

The matrix Γ in this optimization is referred to as the *regularizer*, and introduces the penalty on the recovered signal. While the result x^* may not minimize the residual of the original linear system, it will come close while remaining well behaved (as a result of the influence of the penalty factor).

The solution x^* in (45) can be equivalently defined as

$$x^* = (T^H T + \Gamma^H \Gamma)^{-1} T^H b = (G_T + G_\Gamma)^{-1} T^H b, \quad (46)$$

where G_T and G_Γ are the Gramians of T and Γ , respectively (and it is assumed that the Gramian sum $G = (G_T + G_\Gamma)$ is nonsingular). Given this expression, it is possible to compute x^* with superfast complexity when the matrices T and Γ (or their Gramians) are Toeplitz. In fact, the optimization need not be limited to a single regularization term; the more general problem

$$x^* = \arg \min_x \|Tx - b\|^2 + \|\Gamma_1 x\|^2 + \dots + \|\Gamma_K x\|^2, \quad (47)$$

where the matrices Γ_i form a series of regularizers, can be solved efficiently as well. This more general case has closed-form solution

$$x^* = (G_T + G_{\Gamma_1} + \dots + G_{\Gamma_K})^{-1} T^H b \quad (48)$$

(again assuming that the matrix in this expression is nonsingular). Much as was true for the pseudoinversion of [111], this system can be solved in $\mathcal{O}(\mathbf{N} \log^2 \mathbf{N})$ operations, where \mathbf{N} is the total number of free parameters. It may be further enriched by using the inverse generators of the Gramian sum in an SSD to improve the asymptotic efficiency as the number of columns in b increases.

3.1.2 Regularization algorithm

As presented in Section 2.2.2.1, ET allows block Toeplitz systems to be rewritten in terms of a tangential-interpolation problem. The algorithms and theoretical framework of Section 2.2.3 can then be used to solve these problems in $O(\mathbf{N} \log^2 \mathbf{N})$ operations. To connect these concepts to Tikhonov regularization, it is necessary to reformulate the Tikhonov system of (48) as a block Toeplitz system.

3.1.2.1 Decoupling the Gramians

The developments of Section 2.2.2 that made use of ET for Toeplitz inversion and pseudoinversion required special characteristics of the problems at hand to form the matrix extensions. For inversion, the extension was straightforward since only a single Toeplitz term was involved. For pseudoinversion, since the equation $Tx = b$ formed an overdetermined system, it was necessary to introduce extra conditions to make the system solvable. This goal was achieved by expressing the LS equations in terms of the residual, with one equation $\varepsilon = (b - Tx)$ defining the residual and another expressing its characteristic $T^H \varepsilon = 0$. Together, the equations formed a square, nonsingular block Toeplitz system.

In both prior uses of ET for Toeplitz systems, no manipulation was required to isolate the Toeplitz structure, as only a single Toeplitz matrix was involved in any equation. However, the matrix $G = (G_T + G_{\Gamma_1} + \dots + G_{\Gamma_K})$ that must be inverted to compute (48) does not have such a simple structure. It is composed of an arbitrary number of terms, none of which are necessarily Toeplitz themselves; rather, each term is a product of Toeplitz matrices.² As a result, ET does not directly apply to (48), and there is no convenient residual characterization to use for this problem.

Instead, a general strategy of “decoupling the Gramians” – that is, isolating the terms $\Gamma_k^H \Gamma_k$ into separate equations – may be used to split the individual terms of the Gramian matrices. This manipulation makes it possible to frame the system as a larger set of equations,

²Since G is a sum of products of structured matrices, it can itself be treated as an SSD.

but one that is block Toeplitz. Once the problem has been reframed in this way, ET and the tangential-interpolation algorithm may be applied to compute a solution with superfast complexity.

The Gramian matrices can be decoupled by defining a series of artificial variables $\sigma_0 = Tx^*$, $\sigma_i = \Gamma_i x^*$. Using these variables, the original equation can be re-written as

$$T^H \sigma_0 + \Gamma_1^H \sigma_1 + \cdots + \Gamma_K^H \sigma_K = T^H b,$$

which has replaced terms containing products of Toeplitz matrices with terms containing only a single Toeplitz matrix. It is then also necessary to incorporate the equations defining the relationships of these extra variables to the unknown x^* , yielding the much larger system

$$\begin{bmatrix} \mathbf{0} & T^H & \cdots & \Gamma_K^H \\ T & -\mathbf{I}_{m_0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \Gamma_K & \mathbf{0} & \cdots & -\mathbf{I}_{m_K} \end{bmatrix} \begin{bmatrix} x^* \\ \sigma_0 \\ \vdots \\ \sigma_K \end{bmatrix} = \begin{bmatrix} T^H b \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}. \quad (49)$$

The process of decoupling replaces the original $n \times n$ system with an $|N| \times |N|$ system, where $|N| = n + \sum_i m_i$, with m_i the number of rows in each of the matrices $\{T, \Gamma_1, \dots, \Gamma_K\}$. This new system, however, is block Toeplitz.

As a point of note, regardless of the number of matrices in each row, the extensions will introduce only a single additional artificial variable per block row. For example, supposing $m = \max(m_i)$, the extensions $\widetilde{T}^H, \dots, \widetilde{\Gamma}_K^H$ may be chosen to be $(m+k) \times m_i$ and thus

$$\gamma_0 = \widetilde{T}^H \sigma_0 + \cdots + \widetilde{\Gamma}_K^H \sigma_K.$$

The first block row of the system is then

$$\begin{bmatrix} \mathbf{0} & \begin{bmatrix} \widetilde{T}^H \\ T^H \end{bmatrix} & \cdots & \begin{bmatrix} \widetilde{\Gamma}_K^H \\ \Gamma_K^H \end{bmatrix} & \begin{bmatrix} -\mathbf{I}_{m+k} \\ \mathbf{0} \end{bmatrix} \end{bmatrix} \begin{bmatrix} x^* \\ \sigma_0 \\ \vdots \\ \sigma_K \\ \gamma_0 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ T^H b \end{bmatrix}.$$

Each block column is now truncated circulant, and can be transformed to yield tangential-interpolation conditions.

To forge a clearer connection to the original problem, it is interesting to note that G^{-1} in (48) is the Schur complement of the upper-left zero block of the matrix in (49):

$$G = - \begin{bmatrix} T^H & \dots & \Gamma_K^H \end{bmatrix} \begin{bmatrix} -\mathbf{I}_{m_0} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & -\mathbf{I}_{m_K} \end{bmatrix} \begin{bmatrix} T \\ \vdots \\ \Gamma_K \end{bmatrix}$$

As a consequence, the upper-left corner of the inverse of the matrix in (49) is G^{-1} , resulting in the identity $x^* = G^{-1}T^Hb$. Therefore, in decoupling the Gramians, the original system was replaced by a block Toeplitz system whose inverse has G^{-1} as its $n \times n$ principal leading submatrix. The same general technique can be applied to multi-level Toeplitz, mosaic Toeplitz, and a variety of related systems to allow for the use of ET. However, this will not yield an efficient solution method if the number of block rows and columns is not small relative to the block sizes.

To this point, it has been assumed that each matrix $T, \Gamma_1, \dots, \Gamma_K$ is Toeplitz. If instead the *Gramians* of one or more of these matrices are Toeplitz, then no decoupling is necessary for any such terms. For instance, if T^HT is Toeplitz, it is unnecessary to define σ_0 , saving both a block column and block row from the expanded system.

3.1.2.2 Algorithm summary

The superfast Tikhonov-regularization algorithm may be summarized as follows. First, given matrices T and $\{\Gamma_k\}$, the decoupling variables $\{\sigma_k\}$ are defined and the Gramian components are separated into different block rows. Next, the extension sizes are chosen such that each block row of (49) contains truncated circulant matrices. With the blocks extended, the artificial variables $\{\gamma_k\}$ are defined and the interpolation conditions $\Lambda_{i,j} = \text{diag}(\mathbf{F}C_{i,j}e_1)$, where $C_{i,j}$ is the $(i, j)^{\text{th}}$ block in the extended matrix, are computed. The nodes corresponding to these conditions are the N^{th} roots of unity ω . Only the conditions corresponding to

blocks containing the matrices T and Γ_k or their transposes need to be computed with the FFT, as the others are simply zero or columns from the Fourier matrix. Since the conditions corresponding to the block Γ_k^H can be obtained from those corresponding to the block Γ_k , the interpolation conditions require only K FFTs to compute.

With the interpolation conditions calculated, the τ -degree vector $\tau = [\tau_i]$ is defined as

$$\tau_i = \begin{cases} n - 1 & i = 1 \\ m_{i-2} - 1 & i = 2, \dots, K + 2 \\ N - m_{i-(K+3)} - 1 & i = K + 3, \dots, 2K + 4 \\ 0 & i = 2K + 5 \end{cases}.$$

The τ entries $i = 2, \dots, K + 2$ reflect the degree structure of the artificial decoupling variables σ_k , while the entries $i = K + 3, \dots, 2K + 4$ reflect the degree structure of the artificial extension variables γ_k . The nodes, conditions, and τ vector are then passed to the superfast basis-construction algorithm, which builds a $(2K + 5) \times (2K + 5)$ polynomial basis characterizing the set of all solutions to the interpolation problem. The interpolation is completed in $\mathcal{O}(N \log^2 N)$ operations, as there are $(K + 2)$ sets of N interpolation conditions and it is assumed that $K \ll N$. Once the basis is returned, the column with τ -degree equal to 0 is identified, and the solution x^* is computed by scaling the first element of this column by the last element, which is a scalar.

3.1.3 Displacement and inverse generation

The algorithm of Section 3.1.2 can compute the solution of (48) for an arbitrary vector b in $\mathcal{O}(\mathbf{N} \log^2 \mathbf{N})$ operations. However, since the interpolation conditions that define the problem depend on the entries of b , the solution must be at least partially re-computed for each column of b . Moreover, while the algorithm is capable of applying G^{-1} to a vector, it does not return any explicit characterization that may be used for future applications.

Fortunately, the inverse generators can be used to obtain such a description. To find the displacement structure of G^{-1} , it is necessary to solve a small number of problems

of the form $Gx_i = y_i$ with the algorithm of Section 3.1.2. However, once its generators are known G^{-1} can be written as an SSD whose terms involve circulant and triangular Toeplitz matrices, and thus it may be applied to any vector in $O(n \log n)$ operations with the FFT. This structure provides a considerable computational gain as the number of repeated applications increases.

The generators of the Gramian of a Toeplitz matrix are given by the following theorem.

Theorem 3.1. *Let $T = [a_{i-j}]$ be an $m \times n$ Toeplitz matrix, and define four vectors*

$$u_+ = \begin{bmatrix} 0 \\ a_{1-n} \\ \vdots \\ a_{m-n-1} \end{bmatrix}, \quad u = \begin{bmatrix} 0 \\ a_{1-n} \\ \vdots \\ a_{-1} \end{bmatrix}, \quad r = \begin{bmatrix} 0 \\ \overline{a_{m-1}} \\ \vdots \\ \overline{a_{m-n+1}} \end{bmatrix}, \quad \text{and} \quad r_- = \begin{bmatrix} 0 \\ \overline{a_{m-1}} \\ \vdots \\ \overline{a_1} \end{bmatrix}, \quad (50)$$

where $\overline{a_k}$ is the complex conjugate of a_k . Let e_k be the k^{th} canonical vector of $\mathbb{C}^{m \times m}$, f_k the k^{th} canonical vector of $\mathbb{C}^{n \times n}$, and \hat{x} the reversal of the vector x . Using these notations, define an additional set of vectors

$$\begin{aligned} \hat{s} &= T^T \hat{e}_1 & t &= T^H e_1 \\ v &= T^H (u_+ - T f_1) & \hat{w} &= T^T \hat{r}_-. \end{aligned} \quad (51)$$

Then for $G_T = T^H T$,

$$Z_0 G_T - G_T Z_1 = r \hat{s}^T - t \hat{u}^T + v \hat{f}_1^T - f_1 \hat{w}^T. \quad (52)$$

Proof. See Appendix A. □

Theorem 3.1 gives the rank-4 displacement of the Gramian of a Toeplitz matrix. This result can be generalized to determine the displacement structure of the inverse of a sum of Toeplitz Gramians.

Theorem 3.2. *Given a set $\{T_i\}$ of K Toeplitz matrices, where $T_i \in \mathbb{C}^{m_i \times n}$, assume $G = \sum_i T_i^H T_i$ is nonsingular and let $\{r_i, s_i, t_i, u_i, v_i, w_i\}$ be the corresponding vectors for each*

term $T_i^H T_i$ as defined in (50) and (51). For $\mu = \sum_i v_i$ and $\nu = \sum_i w_i$, define a set of vectors

$$\begin{aligned} \alpha &= G^{-1} f_1 & \hat{\beta} &= G^{-1} \hat{\nu} & \chi &= G^{-1} \mu & \hat{\nu} &= G^{-1} \hat{f}_1 \\ \zeta_i &= G^{-1} t_i & \hat{\xi}_i &= G^{-1} \hat{u}_i & \theta_i &= G^{-1} r_i & \hat{\psi}_i &= G^{-1} \hat{s}_i. \end{aligned}$$

Then

$$Z_1 G^{-1} - G^{-1} Z_0 = \alpha \hat{\beta}^T - \chi \hat{\nu}^T + \sum_{i=1}^K (\zeta_i \hat{\xi}_i^T - \theta_i \hat{\psi}_i^T). \quad (53)$$

Proof. See Appendix A. □

The displacement of G^{-1} from the generalized Tikhonov-regularization problem is given in (53), and can be computed by solving $4(K + 1)$ systems of the form $Gx_i = y_i$ in $\mathcal{O}(\mathbf{N} \log^2 \mathbf{N})$ operations each. Once the solutions to these problems are computed, Theorem 3.2 can be combined with the following result, which gives an SSD of G^{-1} .

Theorem 3.3. *Let the displacement of an $n \times n$ matrix A be given as $Z_1 A - A Z_0 = UV^T$, where $U, V \in \mathbb{C}^{n \times k}$. Then*

$$A = \sum_{j=1}^k C_j L_j,$$

where C_j is an $n \times n$ circulant matrix whose first column is $U f_j$ and L_j is a lower-triangular Toeplitz matrix whose last row is equal to $(V f_j)^T$.

Proof. See [59], Appendix A. □

Theorem 3.3 states that, given the $4(K + 1)$ vectors of the displacement, G^{-1} can be applied with $2(K + 1)$ matrices of the form $C_j L_j$ (referred to hereafter as **C-L products** for brevity). Since the $\{C_j\}$ are circulant, they may be applied with length- n FFTs, while the $\{L_j\}$ may be applied with length- $2n$ FFTs since they are triangular Toeplitz. Therefore, once its $4(K + 1)$ generators are computed, G^{-1} may be applied to any arbitrary vector in only $\mathcal{O}(n \log n)$ operations. When there are $\ell > 4(K + 1)$ columns in b , this leads to a considerable reduction in cost. When $\ell \gg 4(K + 1)$, the overall complexity of the algorithm behaves as $\mathcal{O}(\ell n \log n)$ rather than $\mathcal{O}(\ell \mathbf{N} \log^2 \mathbf{N})$.

3.1.4 Implementation issues

In describing how tangential interpolation can be used to solve Tikhonov-regularization problems, it is worth detailing two previously unexplored implementation issues with the basis-construction algorithm. First is the data-partitioning strategy and its potential to generate many difficult points throughout the progression of the algorithm. Second is the task of constructing a basis when the number of interpolation conditions is not a power of two, as this has the potential to drastically increase the required overhead. By addressing these issues, the algorithm can maintain a small overhead cost across many problem sizes and its accuracy can be further improved.

3.1.4.1 Data partitioning

The first modification to the basis-construction algorithm of [111] is a change in the way the data is subdivided. In other uses of the superfast basis-construction algorithm, the interpolation conditions are subdivided with an interleaving split when the nodes are roots of unity. This approach allows for more efficient basis evaluation, as it reduces the length of the required FFTs in intermediate steps. Unfortunately, it also tends to generate many difficult points for the Tikhonov-regularization problem because of collinearity in the interpolation conditions.

As an example, if Γ_k is $n \times n$, then the expanded block row corresponding to the equation $\sigma_k = \Gamma_k x^*$ is given by

$$\begin{bmatrix} \tilde{\Gamma}_k & \mathbf{0} & \cdots & -\mathbf{I}_{N-n} & \cdots & \mathbf{0} \\ \Gamma_k & -\mathbf{I}_n & \cdots & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}.$$

If the extension matrix $\tilde{\Gamma}_k$ is chosen to be $n \times n$, which will reduce the size of the FFTs in intermediate stages, the interpolation conditions generated by the two identity matrix blocks will be $\{(-1)^k\}$ and $\{-1\}$. After the first subdivision, the two corresponding sets of interpolation conditions will be $\{(-1)^{2k}, -1\}$ and $\{(-1)^{2k+1}, -1\}$. The collinearity between the conditions in each subproblem prevents either from being solved. Since the artificial

variables $\sigma_k(z)$ and $\gamma_{k+1}(z)$ have the same degree structure (they are both of degree $n - 1$) and their interpolation conditions are scaled versions of one another, they become indistinguishable. As a result, most of the points in this subproblem are marked as difficult.

One way to avoid this problem is to modify the subdivision strategy. To be useful, the subdivision strategy must both eliminate predictable collinearity and require short FFT lengths for evaluations. A strategy that satisfies both requirements is “paired-interleaving.” Given conditions $\Phi = \{\phi_1, \phi_2, \dots\}$, the first paired-interleaving data split is

$$\begin{aligned}\Phi^{(1)} &= \{\phi_1, \phi_2, \phi_5, \phi_6, \dots\} \\ \Phi^{(2)} &= \{\phi_3, \phi_4, \phi_7, \phi_8, \dots\}.\end{aligned}$$

Paired interleaving will eliminate collinearity problems arising from identity blocks in the same block row, and will therefore dramatically reduce the number of difficult points identified by the algorithm. Since the difficult points are processed with the fast basis constructor, this corresponds to a dramatic decrease in complexity compared to an interleaving split. Additionally, since difficult points generate numerical instability, this approach tends to be less error-prone.

More importantly, paired interleaving will still allow the algorithm to use small FFTs. While the interleaving has effectively subdivided the data by a factor of four, it requires two sets of conditions to be processed simultaneously. This strategy then requires twice as many FFTs, but the FFTs are half of the size they would be for standard interleaving. As a result, the total number of operations is essentially unchanged.

Even using this technique, however, many difficult points may arise depending on the partitioning of the conditions from each block row. For Toeplitz inversion, there is a single block row in the system and the partitioning is straightforward. The inclusion of the residual characterization results in two block rows in the Toeplitz pseudoinversion problem, and the conditions from these rows may be processed either in isolation or together. In [111], for example, the basis construction for the problem is separated into two stages, each of

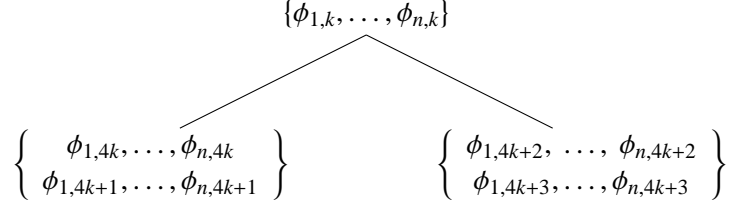


Figure 3: Subdivision process during an iteration of the basis construction. Each set of conditions is subdivided into two components using the paired-interleaving strategy, and corresponding components for each set of conditions are processed together.

which uses a set of the interpolation conditions generated by a single block row.

Extending such an approach to the general Tikhonov problem would result in a final basis computed as the product

$$\mathbf{B}^*(z) = \mathbf{B}_1(z) \cdots \mathbf{B}_c(z),$$

where the bases $\mathbf{B}_i(z)$ are constructed using the conditions generated from the i^{th} block row. Since only one block row contains the solution vector $T^H b$, this type of subdivision saves computation when there are multiple columns in b . Unfortunately, it can also generate many difficult points, and the number of difficult points may vary with the processing order.

A prime example of this behavior is when Γ_k is an ℓ^{th} order difference matrix and the conditions generated by its block row equation $\Gamma_k x^* = \sigma_k$ are processed first. Since no other conditions have been processed, no previous information exists about x^* or σ_k . As a result, there are not nearly enough restrictions on possible solutions to yield a meaningful result. Accordingly, many difficult points will be generated.

Since there is no way of knowing *a priori* what the optimal processing order would be, one may instead consider constructing the basis from the entire set of interpolation conditions simultaneously. To retain the ability to evaluate the subproblem bases efficiently, it is prudent to subdivide the conditions of each block row independently with a paired-interleaving split, as illustrated in Figure 3. This approach tends to be much more efficient for Toeplitz Tikhonov regularization, as it generates many fewer difficult points while retaining the overall subdivision properties of the general algorithm.

3.1.4.2 Data sizes

So far, it has been assumed that the recursive basis-construction algorithm subdivides the conditions until arriving at a single interpolation point. As noted in [110], at some level further subdivision results in higher overhead cost than calling the fast construction routine on the remaining conditions. Therefore, it is prudent to call the fast solver rather than subdivide the total number of interpolation conditions if a given subproblem satisfies $cN_{sub} \leq N_{lim}$ for a machine-dependent N_{lim} (assuming the conditions are processed *en masse*).

If the total number of conditions N is a power of two, there is cancellation in the exponential fractions of the roots of unity during each subdivision. For example, during the first subdivision, the $\{\omega_{2^{k-1}}\}$ are roots of -1 and the basis evaluations $\mathbf{B}_L(\omega_{2^{k-1}})$ are computed with length- $N/2$ FFTs. However, when N is not a power of two, there is no cancellation in the complex exponential factors and longer FFTs are required. As a worst case scenario, when N is prime there will *never* be fractional reduction in the complex exponentials of $\{\omega_{2^{k-1}}\}$. As a result, a full length- N FFT is required to evaluate the basis for each subproblem, regardless of its size. This increased overhead can cause the computation time to vary drastically across similar problem sizes.³

The circulant extensions of Section 2.2.2.1 may be used to circumvent this problem. In the process of extending the Toeplitz matrices in each block row, it is possible to choose $k \geq -1$ such that fractional reduction results in shorter FFTs when evaluating $\mathbf{B}_L(z)$. If $k \geq 0$, it is necessary to choose values for the arbitrary entries in the extension. Empirically, it appears that setting these entries to zero can lead to very ill-conditioned problems. It is therefore wise to choose them to be similar in magnitude to the known matrix coefficients. While there are no theoretical results to justify such an approach, it is more stable

³As an example, when there are $N = 2^{11} = 2048$ conditions there will be cancellation at each subdivision, while if there are $N = 2053$ conditions there will be no cancellation. Despite only containing four more data points, the algorithm performs considerably slower in the latter circumstance.

in practice.

However, the question of how to choose an optimal k remains. One immediate option, adopted in [53], [110], and [111], is to choose k such that N is a power of two (the “power-of-two” method). Unfortunately, this causes the algorithm’s complexity function to become highly quantized relative to the number of interpolation conditions, as there will be twice as many operations required for $N = 2^\ell + 1$ points than for $N = 2^\ell$. Moreover, since the arbitrary values in the extension must be chosen, and there is no *principled* method for doing so, it is best to use the smallest possible extensions.

An alternative strategy, which may also be applied to existing results that make use of the interpolation algorithm, is to impose the following restrictions of the number of interpolation conditions:

1. $N = 2^\ell N_{sub}$, where N_{sub} is an integer;
2. $cN_{sub} \leq N_{lim}$; and
3. $N \geq N_0$, where N_0 is the minimum possible number of conditions in each stage.

The first condition guarantees fractional reduction for the first ℓ subdivisions. The second guarantees that there will be exactly ℓ subdivisions before the fast basis constructor is called. The final condition assures that the extensions indeed produce truncated-circulant blocks (*i.e.*, the circulant extensions are valid). These conditions are devised for an interleaving data split as discussed in Section 2.2.3.2; for a paired-interleaving split, the right side of the inequality in the second condition should be replaced by $N_{lim}/2$ to ensure that the nodes $\{\omega_{4k+i}\}$ can be evaluated efficiently. It is also often more efficient to force N_{sub} to be a product of small primes; this helps to reduce the overhead cost of the FFTs throughout the basis construction.

Algorithm 3 provides a simple method for choosing the extension size k to satisfy these

Algorithm 3 Routine for calculating an alternate circulant extension size k given a minimum number of conditions N_0 and subdivision limit N_{lim} .

procedure $k = \text{ALT_EXTEND}(N_0, N_{lim})$

$N_{sub} \leftarrow N_0$

$N_{tot} \leftarrow cN_{sub}$

$\ell \leftarrow 0$

while $N_{tot} > N_{lim}$ **do**

$\ell \leftarrow \ell + 1$

$N_{sub} \leftarrow \lceil N_{sub}/2 \rceil$

$N_{tot} \leftarrow cN_{sub}$

end while

$k \leftarrow 2^\ell N_{sub} - N_0$

end procedure

criteria. The algorithm is formulated for interleaving data splitting, but can be easily modified for paired-interleaving by changing the **while** loop condition. The number of interpolation conditions processed for the power-of-two method *versus* the proposed alternative method is plotted in Figure 4. For all data points in the curves, sufficient reduction occurs to halve the required FFT length each time the nodes are subdivided. Since the number of conditions is lower, the complexity for the alternative method should increase more smoothly with the problem size. The observed gain in efficiency may also depend on the FFT implementation and the prime factors of N_{sub} .

It is possible to extend this line of thought even further. Rather than a binary subdivision of the data, one could consider subdividing the data at a given stage i into r_i sets, where r_i is an integer factor of N . This method of subdividing the problem will not change the complexity of the algorithm, but may slightly alter the overhead cost of intermediate operations as the number of basis multiplications increases. However, it will also ensure that there is always cancellation in the exponential factors.

The criteria of Algorithm 3 may then be slightly altered. To maximize efficiency, the condition $N = 2^\ell N_{sub}$ may be changed to $N = \prod r_i$, restricting the r_i to be small primes (typically limiting them to the set $\{2, 3, 5, 7\}$). This condition has two important consequences:

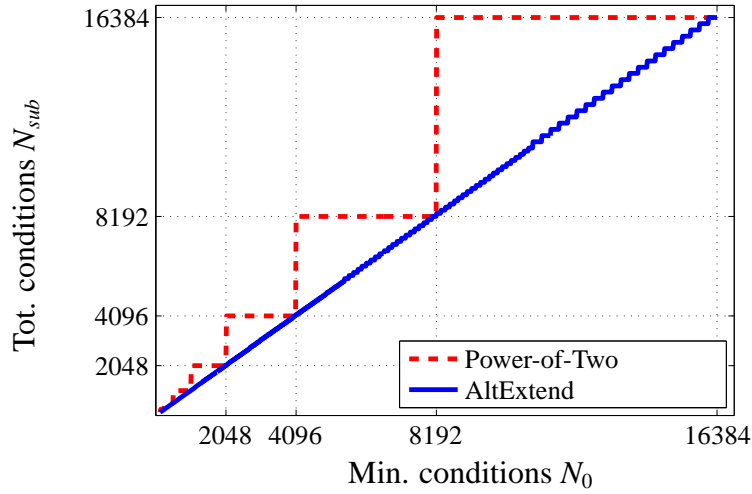


Figure 4: Number of conditions processed versus the minimal possible number of conditions for the power-of-two method and the proposed method of circulant extension, with $N_{lim} = 256$. The “quantization” in the number of conditions processed with the proposed method ALTExtend is significantly milder than that of the power-of-two method.

1. since the size of each r_i is restricted, the number of subdivisions at any stage is small, ensuring the overhead cost does not increase by too much; and
2. since the remaining number of conditions is a product of small primes, all FFTs involved for basis evaluations will be efficient.

The choice of k then amounts to finding the smallest integer such that $N + k = \prod r_i$ with $r_i \in \{2, 3, 5, 7\}$. While this is an integer programming problem, it is small enough in scale that it is fairly inexpensive to solve. Algorithm 4 provides a method for choosing this value of k for a single interpolation set of size N , but requires the recursive basis construction implementation to have flexible subdivision strategies. It may easily be adapted for cases such as Tikhonov regularization which have multiple sets of interpolation conditions.

There is a trade-off between the maximum allowable prime factor r_{max} and the final number of conditions. The larger the allowable factors, the closer the final number of conditions will be to the original number of conditions. However, this comes at the cost of more basis multiplications and less efficient FFTs. When r_{max} is small, Algorithm 4 may result in more interpolation conditions than Algorithm 3, but the overhead cost will

Algorithm 4 Routine for calculating a more optimal circulant extension size k given a minimum number of conditions N_0 and subdivision limit N_{lim} , assuming the recursive tangential interpolation routine subdivides the data based on the prime factors of the data size (rather than just binary divisions).

```

procedure  $k = \text{ALTEXTENDPRIMEFACTORS}(N, r_{max})$ 
   $X \leftarrow 1$ 
   $M \leftarrow N$ 
  for  $r = 2, \dots, r_{max}$  do
    if not  $\text{IsPrime}(r)$  then
      continue
    end if
     $i_{max} \leftarrow \lceil \log_r(N) \rceil$ 
     $X \leftarrow X \otimes r^{(0:i_{max})}$ 
     $M \leftarrow \min_i X_i \text{ s.t. } X_i \geq N$ 
  end for
   $k \leftarrow M - N$ 
end procedure

```

in general be reduced as the FFT sizes are smaller. In practice, however, the restriction to binary subdivisions is sufficient for most problems. It is also worth noting that Algorithm 4 will always result in fewer conditions than Algorithm 3 when $r_{max} = N_{lim}$, but will not necessarily reduce execution time.

3.1.5 Numerical results

The performance of the Tikhonov-regularization algorithm is demonstrated with several numerical simulations. While the algorithm was developed in terms of an arbitrary number of regularizers, these simulations cover cases of practical interest that contain only a single regularizer Γ . In particular, there are two specific classes of problems considered:

- **General problem:** The system matrix T and regularizer Γ are $m_0 \times n$ and $m_1 \times n$ Toeplitz. This class of problems is the most generic in its formulation, encompassing all Toeplitz-structured regularizers for Toeplitz systems. Most notably, this class of problems includes the n^{th} order difference matrices as regularizers, which encourage smoothness in the solutions and penalize signals with significant energy in high frequencies.

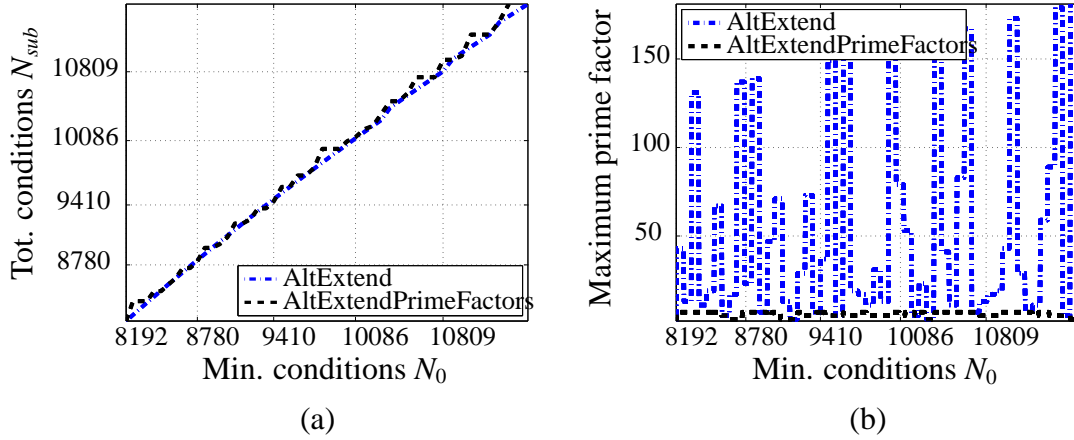


Figure 5: (a) Number of conditions processed versus the minimal possible number of conditions for the ALT_EXTEND and the ALT_EXTENDPRIMEFACTORS methods, with $N_{lim} = 256$ and $r_{max} = 7$. The number of conditions processed in the two methods is similar, with the ALT_EXTENDPRIMEFACTORS method averaging 0.2% more interpolation conditions. (b) Maximum prime factor for the two methods, which is correlated with the efficiency of the FFTs used in intermediate steps. While the ALT_EXTENDPRIMEFACTORS method averages more interpolation conditions, it always involves smaller prime factors and is often preferable to the ALT_EXTEND method.

The decoupled system for the general problem has the form

$$\begin{bmatrix} \mathbf{0} & T^H & \Gamma^H \\ T & -\mathbf{I}_{m_0} & \mathbf{0} \\ \Gamma & \mathbf{0} & -\mathbf{I}_{m_1} \end{bmatrix} \begin{bmatrix} x^* \\ \sigma_0 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} T^H b \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}.$$

Since there are three block rows in the system, three artificial variables are introduced during extension, and the constructed polynomial basis is 7×7 . The matrix $(G_T + G_\Gamma)^{-1}$ has 12 generators and can be applied with six C-L products.

- **Toeplitz-Gramian problem:** In this problem, a single Gramian (either G_T or G_Γ) is itself Hermitian Toeplitz.⁴ This class of problems represents the most common formulation, as it covers standard Euclidean-norm penalization (i.e., Γ is a scaled identity matrix discouraging solutions with large ℓ_2 -norm). In addition, a problem of particular interest within this class is the recovery of a signal from its non-uniform

⁴The regularization amounts to a Toeplitz inversion if both Gramians are Toeplitz, as the sum $G_T + G_\Gamma$ is itself Toeplitz.

Fourier samples, where the matrix T is a non-uniform Fourier matrix and its Gramian is Toeplitz [38]. If the regularizer Γ is also Toeplitz, computing the regularized solution of this problem amounts to solving a Toeplitz-Gramian problem.

Without loss of generality, the decoupled system for the Toeplitz-Gramian problem is

$$\begin{bmatrix} G_T & \Gamma^H \\ \Gamma & -\mathbf{I}_m \end{bmatrix} \begin{bmatrix} x^* \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} T^H b \\ \mathbf{0} \end{bmatrix},$$

where it is assumed that G_T is Hermitian Toeplitz. Two artificial variables are introduced during extension, and the constructed polynomial basis is 5×5 . Using the results of Section 3.1.3, the displacement structure of $G = G_T + G_\Gamma$ is

$$Z_0 G - G Z_1 = (v_2 + u_1) \hat{f}_1^T - f_1 (\hat{w}_2 + \hat{u}_1^T) + r_2 \hat{s}_2^T - t_2 \hat{u}_2^T,$$

where u_1 is the vector u given in (50) for the matrix G_T while the remaining vectors are those of (52) for the matrix Γ . Since there are four terms, the matrix G^{-1} has eight generators and can be applied with four C-L products.

The Toeplitz Tikhonov-regularization algorithm is implemented in C++ with MEX-function interfaces to MATLAB as part of the ‘‘Toeplitz Toolbox’’ code package. All experiments were run on a 3.16 GHz Intel Core 2 Duo machine with 3.0 GB of RAM under the Ubuntu 12.04 LTS operating system and MATLAB R2012A. For each experiment, paired-interleaving data splitting and the extension strategy of Section 3.1.4.2 were employed.

3.1.5.1 Computational complexity

To confirm the algorithm’s asymptotic cost, it was used to solve a large range of Tikhonov problems of varying size. Since the Tikhonov solution x^* is not *exactly* equal to the original input signal x , the algorithm was instead used to compute the solutions to systems of the form $(G_T + G_\Gamma)x = y$ with known input vectors x . These systems are equivalent to

Tikhonov-regularization problems, as y could easily be replaced with $T^H b$, but allow for error comparisons between the calculated x^* and the known inputs.

For each problem type and size, the solution to 1000 systems were computed with the basis-construction algorithm. In each simulation for the general problem, the matrices T and Γ were $n \times n$ Toeplitz with coefficients drawn from a complex standard normal distribution, for a total of $(4n-2)$ free parameters per experiment. For the Toeplitz-Gramian problem, the ℓ_2 -norm-penalization formulation was used, as it is the most common form of Tikhonov regularization. Since there is only a single free parameter in the matrix Γ for this configuration, there were a total of $2n$ free parameters in these experiments. In both cases, the value of n was varied in even logarithmic steps between 2^9 and 2^{15} , with the resulting average execution times plotted in Figure 6. In the figure, the execution times are given as a function of *free parameters* in the problem, rather than as a function of the matrix side length.

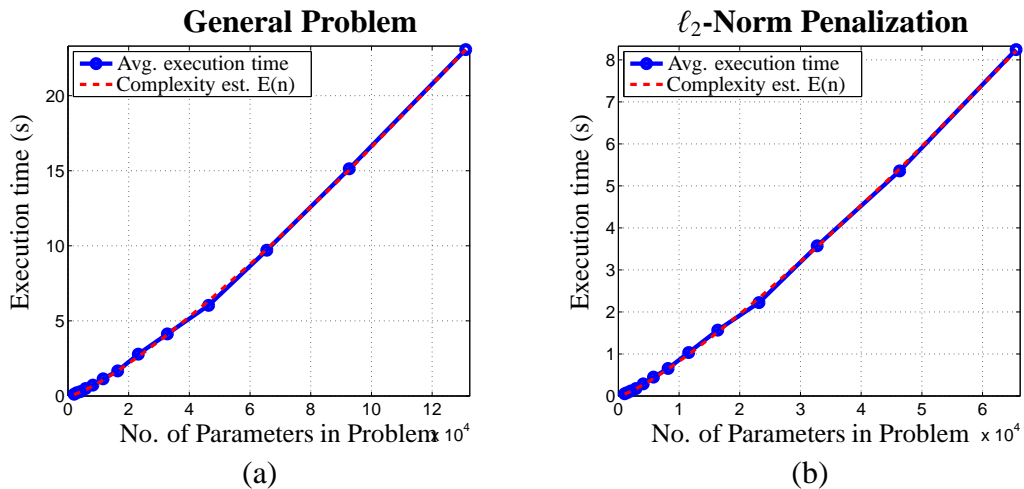


Figure 6: Execution time vs. total number of free parameters for (a) square Toeplitz matrices T and Γ with coefficients drawn from a complex standard normal distribution and (b) square Toeplitz matrices T with coefficients drawn from a complex standard normal distribution and ℓ_2 -norm penalization. For each data point, the execution time was averaged over 1000 trials. Also plotted in the figures are the least-squares estimates of the underlying complexity functions, which closely match the observed performance.

To check the algorithm's scaling properties, the observed execution times are compared to the theoretical underlying cost. Based on the complexity of the component stages of the

algorithm, if the execution time is primarily a function of the number of flops it should be characterized by a function of the form

$$E(n) = \mathcal{C}_1 n \log^2 n + \mathcal{C}_2 n \log n + \text{ov}(n),$$

where \mathcal{C}_1 and \mathcal{C}_2 are constants and $\text{ov}(n)$ reflects the cost of lower-order operations for various problem sizes. For most problems, the execution is dominated by $O(n \log^2 n)$ computations, but given the number of FFTs required the contribution of $O(n \log n)$ calculations is non-negligible for the problem sizes considered. The execution time can then be well-approximated by a function of the form

$$E(n) \approx \mathcal{C}_1 n \log^2 n + \mathcal{C}_2 n \log n.$$

Using this model, a least-squares fit of the constants \mathcal{C}_1 and \mathcal{C}_2 was computed for each class of problem and the estimated computational cost $E(n)$ was superimposed on the observed calculation times in Figure 6.

The execution-time curves in Figure 6 suggest that the performance of the algorithm scales with the predicted asymptotic complexity. The data points in both plots correspond to matrices of the same size. The ℓ_2 -norm-penalization problem requires a significantly smaller computation time than the general problem for systems of the same size. This result is unsurprising given that the Toeplitz-Gramian problem involves fewer interpolation conditions and smaller basis sizes than the general problem. However, when considered as a function of free parameters, there is minimal difference in the interpolator's performance for the two problems. While the general problem requires more operations than ℓ_2 -norm penalization, it also contains roughly twice as many parameters.

3.1.5.2 Accuracy of results

In addition to analyzing the execution time, the results of the experiments of Section 3.1.5.1 were used to verify the accuracy of the implementation of the algorithm. For each experiment, the maximum error between the input vector x and the recovered vector x^* were

Table 1: Maximum error between the input vector and the solution returned from the inversion program for the experiments of Section 3.1.5.1. For each problem type and matrix size, the maximum errors in the recovered vectors were taken across all 1000 trials. “Matrix size” refers to the side length of the matrices for the experiments.

Matrix Size	General Problem	ℓ_2 -Norm Penalization
2^9	9.86E-12	1.55E-11
2^{10}	2.68E-11	4.38E-11
2^{11}	7.56E-11	1.34E-10
2^{12}	1.77E-10	3.79E-10
2^{13}	4.46E-10	1.13E-9
2^{14}	1.19E-9	3.39E-9
2^{15}	2.88E-9	1.07E-8

recorded. The maximum error across all trials was then taken for each type of problem and matrix size.

It is important to note that while the execution time is nearly independent of the matrix conditioning, the accuracy is not. As is to be expected, there are errors that propagate throughout the basis construction. As the matrix $(G_T + G_L)$ becomes more poorly conditioned, the magnitude of these errors increase, resulting in less exact solutions to the interpolation problems. To address the difficulties that this effect might introduce in comparing the algorithm’s accuracy between the two classes of problems, the matrix Γ in the ℓ_2 -norm-penalization experiments was set to $\Gamma = \sqrt{2n}\mathbf{I}_n$, which was determined empirically to yield condition numbers similar to those of the matrices of the general problem. The resulting data better reflects the interpolator’s accuracy when conditions are effectively the same.

The maximum-error results are listed in Table 1, and suggest that the algorithm applies the inverse with an acceptable degree of accuracy for each problem. These results can be further improved with iterative refinement as described in [110]. However, the numbers in Table 1 do not include any such adjustments, and report the errors after a single pass of the inversion process.

Table 2: Scaled maximum error for each element of the vector returned by the basis construction program for the experiments of Section 3.1.5.1. For each problem type and matrix size, the maximum errors in the recovered vectors were taken across all 1000 trials. “Matrix size” refers to the side length of the matrices for the experiments.

Matrix Size	General Problem			ℓ_2 -Norm Penalization	
	$\epsilon(x)$	$\epsilon(\sigma_1)$	$\epsilon(\sigma_2)$	$\epsilon(x)$	$\epsilon(\sigma_1)$
2^9	2.19E-12	6.39E-12	7.08E-12	2.88E-12	2.82E-12
2^{10}	5.20E-12	2.00E-11	1.88E-11	7.54E-12	9.57E-12
2^{11}	9.53E-12	5.23E-11	5.37E-11	1.92E-11	1.61E-11
2^{12}	1.97E-11	1.21E-10	1.21E-10	4.26E-11	4.42E-11
2^{13}	4.12E-11	2.92E-10	3.09E-10	1.28E-10	1.28E-10
2^{14}	9.07E-11	7.84E-10	6.94E-10	2.94E-10	2.81E-10
2^{15}	1.78E-10	1.75E-9	1.76E-9	8.61E-10	6.85E-10

In addition, the errors in the computation of the artificial variables $\sigma_i = T_i^H x$ were recorded. Since the extension sizes vary across problem sizes, these vectors seldom have an ℓ_2 -norm on the same scale as x^* . Thus, for a fair comparison, the maximum error for each of these vectors was scaled by the norm of the true vector. Specifically, the normalized errors are defined as

$$\epsilon(x) = \frac{\max|x - x^*|}{\|x\|}, \quad \epsilon(\sigma_0) = \frac{\max|T^H x^* - \sigma_0^*|}{\|T^H x\|}, \quad \text{and} \quad \epsilon(\sigma_1) = \frac{\max|\Gamma^H x - \sigma_1^*|}{\|\Gamma^H x\|}.$$

The results are shown in Table 2, and demonstrate that the algorithm produces roughly the same degree of error in the artificial variables $\{\sigma_i\}$ as it does for the signal x .

3.1.5.3 Performance for multiple input vectors

To show how displacement structure can speed up the calculation for multiple RHS vectors, the experiments of Section 3.1.5.1 were repeated for y containing multiple columns. In each trial run, the interpolation algorithm was first used to solve for the generators of G^{-1} . With the generators known, the SSD of G^{-1} was used to calculate the solution to the Tikhonov problem for a RHS vector y of size $n \times \ell$. For each matrix size n , the number of columns ℓ was chosen such that the ratio ℓ/n increased in even steps up to 1. The

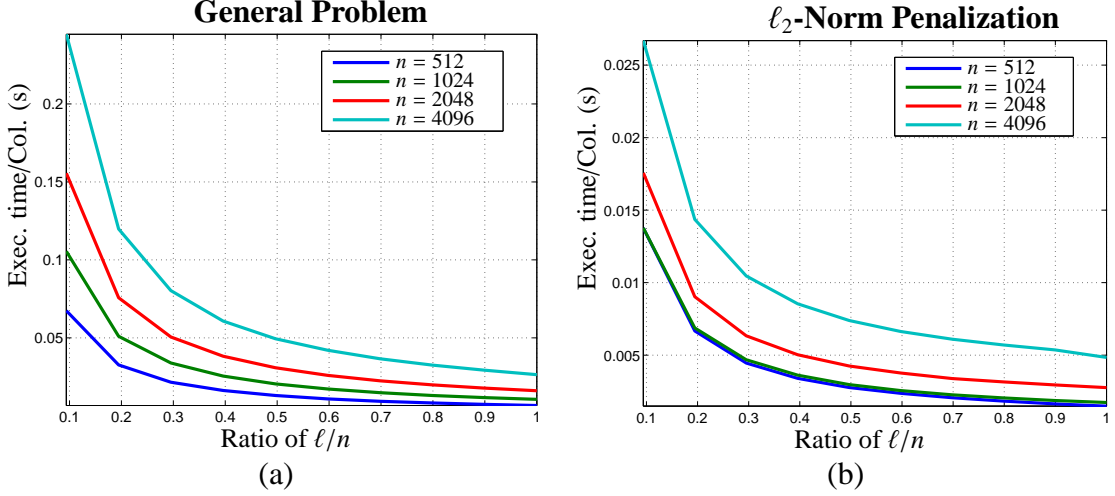


Figure 7: Execution time per column vs. ratio of number of columns in y to the matrix side length n for (a) square Toeplitz matrices T and L with coefficients drawn from a complex standard normal distribution and (b) square Toeplitz matrices T with coefficients drawn from a complex standard normal distribution and ℓ_2 -norm penalization. For each data point, the execution time was averaged over 10 trials, and the inversion program made use of displacement structure to generate the inverse.

per-column execution time of the solver, including the time necessary to compute the generators, was recorded for each size, problem, and ratio. Since the experiments are much more computationally intensive than those of Section 3.1.5.1, the number of trials for each data point was limited to 10 and the range of matrix sizes restricted to $[2^8, 2^{11}]$. The results are shown in Figure 7.

As is evident in the figure, as columns are added to y , the per-column cost of solving the Tikhonov system decreases dramatically. At small ratios, the per-column cost of computing the solution to (48) increases with the matrix size as $\mathcal{O}(N \log^2 N)$, while for larger ratios it increases as $\mathcal{O}(n \log n)$. Therefore, by incorporating displacement structure, the algorithm becomes asymptotically more efficient as the number of columns in y gets larger.

Table 3 lists the maximum errors in each set of trials. Since the accuracy of applying the inverse through its closed-form expression is a function of the accuracy of the computed generators, the errors are slightly larger than when the interpolation is computed for each individual column of y . However, the dropoff is slight compared to the dramatic increase in efficiency. Moreover, it is possible that the numerical stability of the inverse formula could

Table 3: Maximum error between the input vector and the solution returned from the inversion program when using displacement structure to generate the inverse. For each problem type and matrix size, the maximum errors in the recovered solutions were taken across all trials and all ratios of the number of columns ℓ in the input y and the matrix side length n . “Matrix size” refers to the side length of the matrices for the experiments.

Matrix Size	General Problem	ℓ_2 -Norm Penalization
2^7	7.43E-12	1.61E-11
2^8	1.34E-11	9.09E-11
2^9	6.55E-11	3.95E-10
2^{10}	1.34E-10	1.93E-9
2^{11}	5.03E-10	6.61E-9
2^{12}	1.08E-9	2.32E-9

be improved with a different displacement structure (as suggested in [57]).

3.1.5.4 Equivalence in Conjugate Gradient (CG) Iterations

The results of Sections 3.1.5.1 and 3.1.5.2 give an absolute measure of the performance of the tangential interpolator. To gain a sense of perspective, it is useful to translate this performance into a comparison with CG. The choice of comparison with CG is motivated by the fact that there exist no other direct solution methods available for the problems considered, and it is one of the most celebrated iterative methods for solving least-squares problems.⁵ Therefore, it serves as a reasonable benchmark.

A direct comparison with CG is difficult, however. While CG is an iterative method, the tangential-interpolation approach is direct. In addition, the convergence speed of CG is dependent on the conditioning of the matrix (among other factors), resulting in variable solution times across different problems of the same size. By contrast, the tangential-interpolation algorithm is nearly static in computation time for a given matrix size, as the complexity is primarily dependent on the number of interpolation conditions. In spite of these differences, it is possible to compare the relative efficiency of the two algorithms by

⁵Since very large, dense matrices are used in the simulations, any method requiring an explicit construction of the matrices involved will exceed memory capacity for most systems.

determining how many iterations of CG can be performed in the amount of time that the superfast solver requires.

The experiments of Section 3.1.5.1 were repeated and the time required to compute G^{-1} using tangential interpolation in each trial was recorded. The matrix parameters were then passed to an implementation of CG to obtain the solution to the Tikhonov problem, stopping the CG program when the total elapsed time surpassed the direct-inversion time. To arrive at a fair comparison, the CG solver made use of routines designed to apply the matrix G with minimal complexity through FFT operations. In each case, since either the matrices T and L or their Gramians were Toeplitz, the individual matrices could be applied with FFTs of length $3n - 2$. However, the minimum number of FFTs is different for each problem:

- General problem: nine – five to compute Tx and Lx and four to compute $T^H(Tx)$ and $L^H(Lx)$;
- ℓ_2 -norm penalization: five to compute $T^H(Tx)$;

These tallies also use the fact that the FFT of the generating vector of the matrix T^H can be obtained in $O(n)$ operations from the FFT of the generating vector of the matrix T (and similarly for L^H). When the CG programs terminated, the number of *completed* iterations were recorded (discarding the results of any partial iterations), and averaged across all trials. The resulting equivalent iteration counts are given in Table 4.

As indicated in Table 4, the equivalent number of CG iterations increases with the matrix side length. This result is expected, as the CG iterations use $O(n \log n)$ operations while the tangential interpolator is an $O(N \log^2 N)$ algorithm. It is then unsurprising that as n increases more CG iterations can be run in the same amount of time it takes for the tangential interpolator to build a solution.

The accuracy of the two algorithms can also be compared. For each experiment, the maximum error in the solutions returned by the two algorithms across all trials for each

Table 4: Number of CG iterations corresponding to the tangential-interpolation Tikhonov solver for the two Tikhonov-regularization problems. The table values were calculated by averaging the equivalent number of iterations in each scenario over 1000 trials. “Matrix size” refers to the matrix side length for the experiments.

Matrix Size	General Problem	ℓ_2 -Norm Penalization
512	70.0	54.2
1024	74.6	55.7
2048	84.3	60.0
4096	118.6	82.0
8192	126.9	84.7
16384	169.7	104.4
32768	240.2	123.8

Table 5: Maximum errors between the input vector and the returned solutions of both the CG method and the tangential-interpolation algorithm for the two Tikhonov problem types. For each problem type and matrix size, the maximum errors in the recovered vectors were computed across all trials. In all trials, the CG method was terminated after surpassing the time required for the inversion program to return a solution to the same problem. “Matrix size” refers to the side length of the matrices for the experiments.

Matrix Size	General Problem		ℓ_2 -Norm Penalization	
	CG	Direct	CG	Direct
512	1.64E-7	8.66E-12	1.02E-3	1.45E-11
1024	1.61E-7	2.52E-11	5.26E-3	4.45E-11
2048	6.03E-8	7.49E-11	1.21E-2	1.28E-10
4096	1.14E-9	1.70E-10	1.22E-2	4.00E-10
8192	2.02E-10	4.32E-10	1.62E-2	1.14E-9
16384	2.28E-10	1.08E-9	5.85E-2	3.51E-9
32768	1.09E-14	2.75E-9	3.82E-2	1.10E-8

matrix size were recorded. The results are given in Table 5.

Table 5 reflects the potential performance gains that can be realized with the superfast algorithm. The tangential interpolator is only markedly outperformed by CG when the matrices of the general problem become very large. For ℓ_2 -norm penalization, CG requires much more time than the tangential interpolator to achieve a comparable level of accuracy, even when the condition numbers are kept reasonable.

Table 6: Number of CG iterations corresponding to the tangential-interpolation Tikhonov solver using displacement for the general problem. The table values were calculated by averaging the equivalent number of iterations in each scenario over 100 trials. “Ratio” refers to the ratio of the number of columns in the input matrix y to the matrix side length n .

Ratio	$n = 4096$	$n = 8192$
0.11	20.4	18.1
0.37	13.9	5.5
0.53	4.4	3.9
0.79	3.1	2.8
1.0	2.3	2.3

Table 7: Maximum errors between the input vector and the returned solutions of both the CG method and the tangential-interpolation algorithm with displacement for the general problem. For each matrix side length n , the maximum errors in the recovered matrices were computed across all trials and all columns of the input matrix y . In all trials, the CG method was terminated after surpassing the time required for the inversion program to return a solution to the same problem. “Ratio” refers to the ratio of the number of columns in y to the matrix side length n .

Ratio	$n = 4096$		$n = 8192$	
	CG	Direct	CG	Direct
0.11	5.24E0	1.10E-9	5.63E0	2.91E-9
0.37	5.81E0	1.06E-9	6.19E0	3.23E-9
0.53	5.60E0	1.04E-9	5.84E0	2.69E-9
0.79	6.20E0	1.22E-9	5.69E0	3.18E-9
1.0	5.88E0	1.13E-9	5.94E0	2.91E-9

These simulations provide a comparison for problems in which the input y consists of a single column. However, these comparisons can also be made when y has multiple columns. Repeating the experiments for the general problem for different matrix sizes (with a smaller range of sizes since the amount of computation is much larger), the number of columns in y was again varied in fixed ratios up to 1 for various matrix sizes as in Section 3.1.5.3. The equivalent number of CG iterations for two matrix side lengths from these experiments are listed in Table 6 and the corresponding time-limited error comparisons in Table 7.

As is clear from the results, as the number of columns in the RHS increases, the direct

solver dramatically outperforms CG. This result is a product of the fact that the SSD of G^{-1} requires roughly the same number of FFTs as is necessary to apply the matrix G itself. Once the number of columns in the RHS is larger than the number of inverse generators, the equivalent number of CG iterations necessary to process additional columns in y becomes very small.

3.1.5.5 *Non-uniform Fourier samples*

To demonstrate a practical use of the algorithm, it was applied to the task of reconstructing a signal from non-uniform spectral samples. This is a common problem in signal processing, and one that is particularly relevant (albeit in a multi-dimensional variant) for magnetic resonance imaging (MRI). For one-dimensional signals, the spectrum of a discrete signal $x = [x_j]$ may be sampled at an arbitrary frequency $f_0 \in [-1/2, 1/2)$ by evaluating the sum

$$X(f_0) = \sum_j x_j \mathbf{e}^{-j2\pi f_0 j}.$$

Collectively, a set of K spectral samples at frequencies $\{f_k\}$ may be obtained by evaluating the matrix-vector product $X = Ax$, where A is a Fourier-like matrix with entries

$$A_{kj} = \mathbf{e}^{-j2\pi f_k j}.$$

When $K = N$ and the $\{f_k\}$ are uniformly spaced in $[-1/2, 1/2)$, the matrix A is a Fourier matrix. Accordingly, the signal x may be recovered from its spectral samples by

$$A^H X = A^H A x = x.$$

However, when the frequencies are not uniformly spaced, the matrix A is often severely ill-conditioned, and a regularized solution to the system $X = Ax$ is required. The Tikhonov regularization for the problem is typically formulated as

$$\hat{x} = (A^H W A + \Gamma^H \Gamma)^{-1} A^H W X,$$

where W is a diagonal weighting matrix that compensates for the sampling density in the Fourier domain and Γ is the regularizer. For MRI reconstruction problems, a Voronoi-cell weighting is typically used to compute the entries of W [90].

Straightforward calculations show that the “weighted Gramian” A^HWA is structured, with entries

$$(G_A)_{i,j} = (A^HWA)_{i,j} = \sum_{k=1}^K w_{kk} e^{j2\pi f_k(i-j)}.$$

Since the entries of G_A depend only on the index difference $(i - j)$, G_A is Toeplitz. If the Tikhonov matrix Γ is also Toeplitz, this amounts to the Toeplitz-Gramian problem, and the tangential-interpolation algorithm may be used to determine the solution to the Tikhonov-regularization problem.

To demonstrate the use of tangential interpolation for this problem, length-4096 input signals consisting of linear combinations of sinusoids of three randomly-chosen frequencies in the digital frequency range $[0, 0.02]$ were generated. For each input signal, 4096 spectral samples at random frequencies chosen from a triangular distribution over $[-1/2, 1/2)$ were computed. As the number of Fourier samples was equal to the length of the input signals, the corresponding Gramian matrices G_A were severely rank-deficient.

Since the input signals contained only low-frequency sinusoids, a scaled second-order difference matrix

$$\Gamma_{i-j} = \begin{cases} -1\text{E-}4 & |i - j| = 1 \\ 2\text{E-}4 & i = j \\ 0 & \text{else} \end{cases}$$

served as a mild but effective regularizer, penalizing solutions with high-frequency content.

A typical reconstruction is shown in Figure 8. In this example, the Gramian G_A had a numerical rank of 3050 despite being 4096×4096 , and the condition number of the matrix $(G_A + \Gamma^H\Gamma)$ was approximately $9.7\text{E}6$. Despite this poor conditioning, a reasonable reconstruction was acquired through direct inversion in less than 0.7 seconds.

The reconstruction may be compared to one achieved with CG. Again time-limiting the CG reconstruction based on the runtime of the tangential interpolator, the iterative solution to the same problem was computed. With both solutions available, the error vector $(x - x^*)$ for each method was computed and the results plotted on the same scale in Figure 9. While

Nonuniform Fourier Reconstruction (4096 Samples)

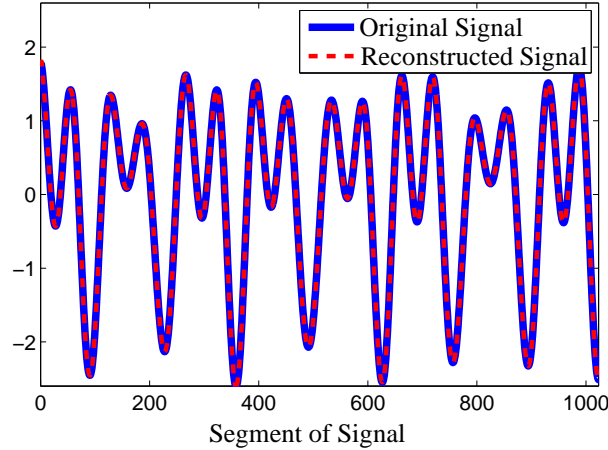


Figure 8: A segment of the reconstruction of a low-frequency input signal using 4096 non-uniformly spaced Fourier samples via superfast Tikhonov regularization, where the Tikhonov matrix was a scaled second-order difference matrix. Despite the poor conditioning of the matrix, a reasonable reconstruction was produced in under a second.

both methods achieve reasonable reconstructions, the tangential-interpolation algorithm outperforms CG in reconstruction quality for an equal amount of computation time.

3.2 Nonuniform resampling of digital signals

Resampling discrete signals is one of the fundamental operations in digital signal processing. For an analog signal $x(t)$ sampled at a period T above the Nyquist rate, classical signal processing theory states that $x(t)$ can be synthesized from its samples $x_j = x[j] = x(jT)$ by

$$\widehat{x}(t) = \sum_{j \in \mathbb{Z}} x[j] \cdot \text{sinc}(t/T - j),$$

where $\text{sinc}(x) = \sin(\pi x)/(\pi x)$. During the process of digital resampling, the signal is not actually synthesized to an analog version and then sampled again. Instead, new samples are calculated at the desired locations σ_i through the synthesis formula

$$x_{\text{nu}}[i] = \sum_{j \in \mathbb{Z}} x[j] \cdot \text{sinc}(\sigma_i - j). \quad (54)$$

The synthesis formula of (54) is an interpolation, where the input samples $x[j]$ are weighted by samples of the sinc kernel function.

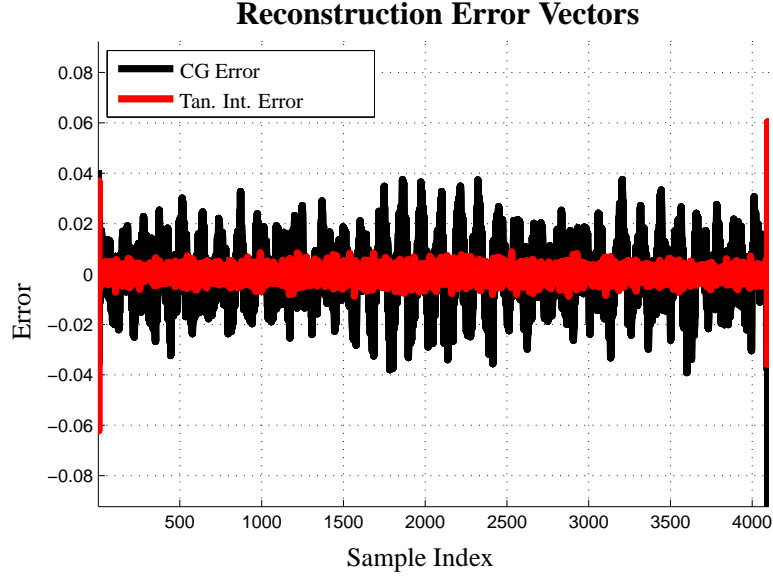


Figure 9: Errors of the reconstructed signals from the tangential-interpolation method and the CG method. Both methods yield a reasonable reconstruction in the same amount of time, with the tangential-interpolation method having a smaller error level.

When the new sampling locations σ_i form a uniform grid, this process amounts to altering the sampling rate of the signal. Such problems are well-studied, and the computation involved is fairly straightforward. For a rational rate change by a factor of P/Q (or an irrational rate change suitably approximated by a rational rate change), the new samples are calculated by the following steps:

1. A longer, zero-padded version of the signal is created:

$$\tilde{x}_{i+jP} = \begin{cases} x_j & i = 0 \\ 0 & i \neq 0 \end{cases}$$

for $i = 0, \dots, P - 1$ and $j = 0, \dots, n - 1$.

2. The signal \tilde{x} is filtered with an anti-aliasing filter $\widehat{x} = T\tilde{x}$. This operation corresponds to multiplication by a Toeplitz matrix whose entries are determined by the filter coefficients.
3. The new signal is calculated as a decimated version of the filtered signal, $y_i = \widehat{x}_{Qi}$.

The only real computation involved is in the filtering stage, where a Toeplitz matrix-vector multiplication is performed.

For many applications (including packet-data traffic, geophysics, medical imaging, and astronomy, for example), either the input or output sample grids are *non-uniform*, and the resampling process is used to convert between uniform and non-uniform samples [33, 100]. There is a great deal of literature on the theoretical properties of non-uniform resampling [18, 23, 35, 100] and many algorithms for converting between sampling modalities [11, 36, 74, 114]. These algorithms are diverse, with some iterative and others direct, some approximative and others exact, *etc.*

It is also possible to express the non-uniform resampling problem in terms of structured matrices by exploiting the behavior of the sinc interpolation kernel. Since the uniform resampling problem uses only structured matrices in its computational step, this approach allows for a more unified view of resampling in terms of structured linear algebra. Moreover, by exploiting the structure of the problem, it is possible to obtain asymptotically efficient algorithms for the sinc interpolation involved in resampling.

3.2.1 Structured matrices in non-uniform resampling

Expressing the resampling problem as a linear system, a digital signal $x \in \mathbb{C}^n$ can be resampled at arbitrary locations through the matrix-vector multiplication $\widehat{x} = Rx$, where $R \in \mathbb{C}^{m \times n}$ has entries

$$R_{i,j} = \text{sinc}(\sigma_i - j).$$

Using basic trigonometric manipulations, the elements of R satisfy

$$\begin{aligned} R_{i,j} &= \frac{\sin(\pi(\sigma_i - j))}{\pi(\sigma_i - j)} = \frac{\sin(\pi\sigma_i) \cos(\pi j) - \sin(\pi j) \cos(\pi\sigma_i)}{\pi(\sigma_i - j)} \\ &= \frac{\sin(\pi\sigma_i)(-1)^j}{\pi(\sigma_i - j)} \end{aligned}$$

Define the following variables:

$$\begin{aligned} c_i &= \pi\sigma_i & d_j &= \pi j \\ z_i &= \sin(c_i) & y_j &= (-1)^j. \end{aligned}$$

If C is the Cauchy matrix $C_{i,j} = (c_i - d_j)^{-1}$, one can write R as the generalized Cauchy matrix $R = \text{diag}(z) \cdot C \cdot \text{diag}(y)$. Since R is a generalized Cauchy matrix with $r = 1$, it can be applied with the FMM in $\mathcal{O}(m + n)$ operations. This procedure is shown in Algorithm 5.

Algorithm 5 Algorithm to resample a uniform signal $x[k]$, $k = 0, \dots, n - 1$ onto the non-uniform nodes σ_i using sinc interpolation.

procedure $\widehat{x} = \text{UNI_TO_NON_UNI_RESAMPLE}(x, \sigma)$

$N \leftarrow \text{LENGTH}(x)$

$c_i \leftarrow \pi\sigma_i$

$d_j \leftarrow \pi j$

$\alpha_j \leftarrow x_j \cdot (-1)^j$

$y \leftarrow \text{FMM}(\alpha, c, d)$

$\widehat{x}_i \leftarrow \sin(\pi\sigma_i) \cdot y_i$

end procedure

For most practical applications, however, the problem at hand is typically to obtain uniform samples x from non-uniform samples \widehat{x} rather than *vice versa*. This type of resampling may be viewed as the inverse problem of computing Rx ; that is, computing $x = R^\dagger \widehat{x}$, where $(\cdot)^\dagger$ denotes the pseudoinverse. In general, the problem is only solvable and well-conditioned if the matrix R is tall and has full column rank. In this case, the least-squares solution is

$$x^* = R^\dagger \widehat{x} = (R^T R)^{-1} R^T \widehat{x} = G^{-1} R^T \widehat{x}.$$

It is generally expensive to compute x^* , as the usual cost of solving a system $G\mu = \nu$ is $\mathcal{O}(n^3)$. However, G is also structured, and its displacement allows for an efficient inversion.

3.2.2 Superfast non-uniform-to-uniform resampling

The matrix G of Section 3.2.1 is the Gramian of a generalized Cauchy matrix. Letting $D = \text{diag}(\pi_j)$, the displacement of G is

$$\nabla_{D,D}(G) = uv^T - vu^T,$$

where $u = R^T z$ and $v = y$ as prescribed by Corollary 2.6. Using (36), the matrix G can be reconstructed with the pseudoinverse of the displacement operator from its generators u and v and nodes d_j up to its main diagonal :

$$[\nabla^\dagger(u, v, d)]_{i,j} = \begin{cases} \frac{u_i v_j - v_i u_j}{\pi(i-j)} & i \neq j \\ 0 & \text{else} \end{cases}.$$

The matrix $\nabla^\dagger(u, v, d)$ cannot be expressed as a generalized Cauchy matrix, as the denominators of its diagonal coefficients are undefined. However, the uniformity of the nodes $\{\pi_j\}$ still allows for fast computation. Define a ‘‘core’’ Toeplitz matrix

$$T_{i,j} = \begin{cases} \alpha & (i-j) = 0 \\ \frac{1}{\pi(i-j)} & \text{else} \end{cases},$$

where α is any scalar, and diagonal matrices $D_u = \text{diag}(u)$ and $D_v = \text{diag}(v)$. Then the pseudoinverse of the displacement operator has the SSD $\nabla^\dagger(u, v, d) = D_u T D_v - D_v T D_u$, and may thus be applied in $\mathcal{O}(n \log n)$ operations. In practice, setting $\alpha = 1$ yields a matrix T with both condition number and operator norm close to 1.

If the main diagonal λ of G is also known, then the product $G\mu$ may be calculated with the SSD

$$G\mu = (D_u T D_v - D_v T D_u + \text{diag}(\lambda))\mu.$$

This identity is also useful in determining the diagonal of G^{-1} once its generators are known. Denoting the inverse generators as $p = G^{-1}v$ and $q = G^{-1}u$, let $D_p = \text{diag}(p)$

and $D_q = \text{diag}(q)$. Designate the main diagonal of G^{-1} as γ and let $\mu = \beta_1 v + \beta_2 u$ and $v = \beta_1 p + \beta_2 q$ for some scalars $\beta_1, \beta_2 \in \mathbb{C}$; then

$$\begin{aligned}
v &= G^{-1}\mu = (D_p T D_q - D_q T D_p)\mu + \text{diag}(\gamma)\mu \\
&= (D_p T D_q - D_q T D_p)\mu + \text{diag}(\mu)\gamma \\
\Rightarrow \gamma &= \text{diag}(\mu)^{-1} \left(v - (D_p T D_q - D_q T D_p)\mu \right). \tag{55}
\end{aligned}$$

All operations in (55) involve diagonal or Toeplitz-matrix multiplies. Therefore, once the generators of G^{-1} are known, its main diagonal may be found in $O(n \log n)$ operations. With the main diagonal calculated, G^{-1} may be applied to any vector in $O(n \log n)$ operations with similar operations.

The remaining question is how to compute the generators p and q . Given that the matrix G cannot be fully reconstructed from its displacement alone, Algorithm 2 must be modified. Specifically, the main diagonal of the matrix inverses in the recursion must also be computed to fully recover G^{-1} . If the diagonal of G is supplied, the diagonal of the upper-left block $G_{1,1}$ is known. The first recursive call is then unchanged if the main diagonal of the matrix G is also an input parameter to the algorithm.

However, the diagonal of the Schur complement S must also be computed. The diagonal of S can be found using (55) once the generators u_S and v_S are computed. More specifically, if λ_S is the main diagonal of the Schur complement, then

$$\begin{aligned}
\lambda_S &= S\mathbf{1} - \nabla^\dagger(u_S, v_S, d_2)\mathbf{1} \\
&= G_{2,2}\mathbf{1} - G_{1,2}^T G_{1,1}^{-1} G_{1,2}\mathbf{1} - \nabla^\dagger(u_S, v_S, d_2)\mathbf{1}.
\end{aligned}$$

Once λ_S is known, the second recursive call can be made as usual. The final step of the algorithm is to compute the diagonal of G^{-1} from (55) for some selected values β_1 and β_2 . The full modified algorithm is given in Algorithm 6, where the generators are defined slightly differently to exploit the symmetry of G . Namely, since G^{-1} is symmetric, it is only necessary to compute the left generators $G^{-1} \begin{bmatrix} u & v \end{bmatrix}$ since the right generators $\begin{bmatrix} v & -u \end{bmatrix}^T G^{-1}$

are identical.

Algorithm 6 A recursive algorithm to obtain the generators of G^{-1} , the inverse of the Gramian of a resampling matrix. The algorithm partitions G^{-1} and computes its generators and main diagonal by computing the generators and main diagonal of the inverse of the principal leading submatrix $G_{1,1}$ and of the inverse of the Schur complement S . The matrices $G_{i,j}$ and S are applied using their displacements.

procedure $[p, q, \gamma] = \text{RESAMPGRAMINVGEN}(u, v, \lambda, d)$

$N \leftarrow \text{LENGTH}(u)$

if $N < N_{min}$ **then**

$G_{i,j} \leftarrow \nabla^\dagger(u, v, d) + \text{diag}(\lambda)$

$p \leftarrow G^{-1}v$

$q \leftarrow G^{-1}u$

else

$[\tilde{p}_1, \tilde{q}_1, \tilde{\gamma}_1] \leftarrow \text{RESAMPGRAMINVGEN}(u_1, v_1, \lambda_1, d_1)$

$u_S \leftarrow u_2 - G_{1,2}^T \tilde{q}_1$

$v_S \leftarrow v_2 - G_{1,2} \tilde{p}_1$

$\lambda_S \leftarrow (S - \nabla^\dagger(u_S, v_S, d_2))\mathbf{1}$

$[p_2, q_2, \gamma_2] \leftarrow \text{RESAMPGRAMINVGEN}(u_S, v_S, \lambda_S, d_2)$

$p_1 \leftarrow \tilde{p}_1 - G_{1,1}^{-1} G_{1,2} p_2$

$q_1 \leftarrow \tilde{q}_1 - G_{1,1}^{-T} G_{1,2} q_2$

end if

$\mu \leftarrow \beta_1 v + \beta_2 u$

$v \leftarrow \beta_1 p + \beta_2 q$

$\gamma \leftarrow \text{diag}(\mu)^{-1} (v - \nabla^\dagger(p, q, d)\mu)$

end procedure

There are two appreciable differences between Algorithms 2 and 6. First, Algorithm 6 requires extra computation to compute the diagonal values. However, all matrix operations involved require $O(n \log n)$ or $O(n)$ operations, so the asymptotic efficiency of the overall algorithm is unchanged. Second, from the symmetry of the matrix G , the amount of computation is halved since the two generator matrices contain the same vectors.

3.2.3 Efficient pseudoinversion for resampling matrices

Algorithm 6 gives a straightforward approach to finding the generators of G^{-1} , but much of the calculation can be eliminated when it is taken in the broader context of computing the

pseudoinverse of R . The displacement of R^\dagger is

$$\begin{aligned}
\nabla_{D_d, D_c}(R^\dagger) &= \nabla_{D_d, D_c}(G^{-1}R^T) = \nabla_{D_d, D_d}(G^{-1})R^T + G^{-1}\nabla_{D_d, D_c}(R^T) \\
&= pq^T R^T - qp^T R^T - G^{-1}yz^T \\
&= pq^T R^T - qp^T R^T - pz^T \\
&= \begin{bmatrix} p & q \end{bmatrix} \begin{bmatrix} Rq - z \\ -Rp \end{bmatrix}^T = \chi Y^T.
\end{aligned}$$

If $c_i \neq d_j$ for all i, j , then the displacement $\nabla_{D_d, D_c}(\cdot)$ is nonsingular. Therefore, there is no actual need to compute the main diagonal of G^{-1} ; if the generators p and q are known, the generators of R^\dagger may be computed without needing to apply G^{-1} directly.

Similar logic can be applied throughout the recursive procedure as well. Suppose

$$\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} G_{1,1} & G_{1,2} \\ G_{1,2}^T & G_{2,2} \end{bmatrix} \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \end{bmatrix}.$$

Then

$$\begin{aligned}
\delta_2 - G_{1,2}^T G_{1,1}^{-1} \delta_1 &= G_{1,2}^T \mathbf{1} + G_{2,2} \mathbf{1} - G_{1,2}^T G_{1,1}^{-1} G_{1,1} \mathbf{1} - G_{1,2}^T G_{1,1}^{-1} G_{1,2} \mathbf{1} \\
&= (G_{2,2} - G_{1,2}^T G_{1,1}^{-1} G_{1,2}) \mathbf{1} = S \mathbf{1}.
\end{aligned}$$

Therefore, if the quantity $\delta = G \mathbf{1}$ is known, the product $S \mathbf{1}$ can be computed with an application of the matrix $G_{1,2}^T G_{1,1}^{-1}$.

The generators of the product $G_{1,2}^T G_{1,1}^{-1}$ are

$$\begin{aligned}
\nabla_{D_{d_2}, D_{d_1}}(G_{1,2}^T G_{1,1}^{-1}) &= \nabla_{D_{d_2}, D_{d_1}}(G_{1,2}^T) G_{1,1}^{-1} + G_{1,2}^T \nabla_{D_{d_2}, D_{d_1}}(G_{1,1}^{-1}) \\
&= (u_2 v_1^T - v_2 u_1^T) G_{1,1}^{-1} + G_{1,2}^T (\tilde{p}_1 \tilde{q}_1^T - \tilde{q}_1 \tilde{p}_1^T) \\
&= (u_2 \tilde{p}_1^T - v_2 \tilde{q}_1^T) + G_{1,2}^T (\tilde{p}_1 \tilde{q}_1^T - \tilde{q}_1 \tilde{p}_1^T) \\
&= (u_2 - G_{1,2}^T \tilde{p}_1) \tilde{p}_1^T - (v_2 - G_{1,2}^T \tilde{q}_1) \tilde{q}_1^T \\
&= u_S \tilde{p}_1^T - v_S \tilde{q}_1^T.
\end{aligned}$$

Since the nodes of d_1 and d_2 are disjoint sets of d , they are mutually exclusive. Therefore, $G_{1,2}^T G_{1,1}^{-1}$ can be expressed in an SSD that uses its generators. Defining the Toeplitz matrix

$$(\tilde{T}_1)_{i,j} = \frac{1}{(d_2)_i - (d_1)_j},$$

it follows that

$$\begin{aligned} G_{1,2}^T G_{1,1}^{-1} &= \tilde{\mathbf{V}}^{-1} \left(\begin{bmatrix} u_S, v_S \end{bmatrix}, \begin{bmatrix} \tilde{p}_1 & -\tilde{q}_1 \end{bmatrix}, d_2, d_1 \right) \\ &= \text{diag}(u_S) \tilde{T}_1 \text{diag}(\tilde{p}_1) - \text{diag}(v_S) \tilde{T}_2 \text{diag}(\tilde{q}_1). \end{aligned}$$

Given δ , then, it is possible to compute the main diagonal of S with two generalized Cauchy matrix-vector products rather than the six products required in Algorithm 6. Similarly, for the computation of p_1 and q_1 , the matrix $G_{1,1}^{-1} G_{1,2} = (G_{1,2}^T G_{1,1}^{-1})^T$ has generators

$$\nabla_{D_{d_1}, D_{d_2}}(G_{1,1}^{-1} G_{1,2}) = \tilde{q}_1 v_S^T - \tilde{p}_1 u_S^T.$$

Therefore, only two (rather than four) generalized Cauchy matrix-vector products are needed to compute p_1 and q_1 .

Since $G_{1,1}^{-1}$ is only used in conjunction with $G_{1,2}$ or $G_{1,2}^T$, its main diagonal is never needed. Similarly, it is unnecessary to compute the main diagonal for S^{-1} , as it is never used. Finally, since the pseudoinverse itself does not require the main diagonal of G^{-1} , there is no need to *ever* compute the main diagonal during the recursive call. The main diagonals are only used in the beginning of the algorithm to compute $G\mathbf{1}$. Thus, the diagonal of the matrix to be inverted must be supplied, but the diagonal of its inverse never needs to be computed.

This more economical approach is given in Algorithm 7, which is used to provide the generators of R^\dagger rather than G^{-1} . While the matrix G is more expensive to apply than any of its individual blocks, by computing $G\mathbf{1}$ the number of matrix multiplies required in various stages of the algorithm is greatly reduced. Moreover, the products $G_{1,1}\mathbf{1}$ and $S\mathbf{1}$ are

computed throughout, and can be passed to the recursive routine to avoid further calculation.⁶ Therefore, this approach significantly reduces overhead and eliminates redundant calculations.

With this algorithm established, Algorithm 8 then provides a superfast non-uniform-to-uniform resampling. The first lines determine the generators u and v of G , as well as its main diagonal. Then, the generators of the pseudoinverse are determined with Algorithm 7. Finally, the pseudoinverse generators are used in an SSD to multiply the pseudoinverse by the non-uniform samples. Note that it is also possible to incorporate the procedure of performing the multiplication of the input samples directly into the routine of Algorithm 7.

3.2.4 Performance scaling

The performance scaling of an implementation of the resampling algorithms can be observed with a series of numerical simulations. For these simulations, the algorithms were implemented in C++ with MEX-function interfaces to MATLAB. All experiments were run on a 3.16 GHz Intel Core 2 Duo machine with 3.0 GB of RAM under the Ubuntu 12.04 LTS operating system and MATLAB R2012A.

In the simulations, a number of input signals of varying lengths were processed with the resampling algorithms and the accuracy and execution time was recorded. More specifically, the size of the uniform sampling grid was varied between 2^8 and 2^{15} in even logarithmic steps. Taking the uniform grid size to be n , in each of the 10000 trials per grid size a non-uniform sampling grid was then formed by choosing $m = 3n$ sampling points at random in the interval $(0, n - 1/2)$.

To estimate the accuracy of the algorithms, the results of the resampling algorithms were compared with direct calculations. Since the direct computation of sinc interpolants is computationally costly at $O(mn)$, it is beneficial to use s sparse signals (signals with only

⁶Since $\delta_1 = G_{1,1}\mathbf{1} + G_{1,2}^T\mathbf{1}$, it is necessary to pass the value $\hat{\delta}_1 = \delta_1 - G_{1,2}^T\mathbf{1}$ to the recursive procedure on the block $G_{1,1}$.

Algorithm 7 A recursive algorithm to obtain the generators of R^\dagger , the pseudoinverse of a non-uniform resampling matrix. The algorithm operates by computing the generators of $(R^T R)^{-1}$ through a partitioning and then uses them to compute the generators of the full matrix $(R^T R)^{-1} R^T$. By never explicitly computing the diagonal of G^{-1} , redundant calculation is eliminated. The $\tilde{\nabla}$ operators are used to apply generalized Cauchy blocks with nonsingular displacements with their SSDs, while the ∇^\dagger operators are used to apply matrices with low displacement rank but singular displacements.

procedure $[\chi \quad \Upsilon] = \text{RESAMPSEUDOINVGEN}(z, y, c, d)$

$$u \leftarrow \tilde{\nabla}^{-1}(y, -z, d, c)z$$

$$v \leftarrow y$$

$$\delta \leftarrow \left(\tilde{\nabla}^{-1}(y, -z, d, c) \right) \left(\tilde{\nabla}^{-1}(z, y, c, d) \right) \mathbf{1}$$

▷ Compute $\delta = R^T R \mathbf{1}$

$$\lambda \leftarrow \delta - \nabla^\dagger(u, v, d) \mathbf{1}$$

▷ Compute main diagonal of G

$$\begin{bmatrix} p & q \end{bmatrix} = \text{RECGRAMINVGEN}(u, v, \lambda, \delta, d)$$

$$\chi \leftarrow \begin{bmatrix} p & q \end{bmatrix}$$

$$\Upsilon \leftarrow \tilde{\nabla}^{-1}(z, y, c, d) \begin{bmatrix} q & -p \end{bmatrix} - \begin{bmatrix} z & \mathbf{0} \end{bmatrix}$$

end procedure

procedure $\begin{bmatrix} p & q \end{bmatrix} = \text{RECGRAMINVGEN}(u, v, \lambda, \delta, d)$

$$N \leftarrow \text{LENGTH}(u)$$

if $N < N_{min}$ **then**

$$G_{i,j} \leftarrow \nabla^\dagger(u, v, d) + \text{diag}(\lambda)$$

$$\begin{bmatrix} p & q \end{bmatrix} \leftarrow G^{-1} \begin{bmatrix} v & u \end{bmatrix}$$

else

$$\hat{\delta}_1 \leftarrow \delta_1 - \tilde{\nabla}^{-1} \left(\begin{bmatrix} u_1 & v_1 \end{bmatrix}, \begin{bmatrix} v_2 & -u_2 \end{bmatrix}, d_2, d_1 \right) \mathbf{1}$$

▷ Update δ for $G_{1,1}$

$$\begin{bmatrix} \tilde{p}_1 & \tilde{q}_1 \end{bmatrix} \leftarrow \text{RECGRAMINVGEN}(u_1, v_1, \lambda_1, \hat{\delta}_1, d_1)$$

$$\begin{bmatrix} u_S & v_S \end{bmatrix} \leftarrow \begin{bmatrix} u_2 & v_2 \end{bmatrix} - \tilde{\nabla}^{-1} \left(\begin{bmatrix} u_2 & v_2 \end{bmatrix}, \begin{bmatrix} v_1 & -u_1 \end{bmatrix}, d_2, d_1 \right) \begin{bmatrix} \tilde{q}_1 & \tilde{p}_1 \end{bmatrix}$$

$$\delta_S \leftarrow \delta_2 - \tilde{\nabla}^{-1} \left(\begin{bmatrix} u_S & v_S \end{bmatrix}, \begin{bmatrix} \tilde{p}_1 & -\tilde{q}_1 \end{bmatrix}, d_2, d_1 \right) \delta_1$$

▷ Compute $\delta = S \mathbf{1}$

$$\lambda_S \leftarrow \delta_S - \nabla^\dagger(u_S, v_S, d_2) \mathbf{1}$$

▷ Find main diagonal of S

$$\begin{bmatrix} p_2 & q_2 \end{bmatrix} \leftarrow \text{RECGRAMINVGEN}(u_S, v_S, \lambda_S, \delta_S, d_2)$$

$$\begin{bmatrix} p_1 & q_1 \end{bmatrix} \leftarrow \begin{bmatrix} \tilde{p}_1 & \tilde{q}_1 \end{bmatrix} - \tilde{\nabla}^{-1} \left(\begin{bmatrix} \tilde{q}_1 & \tilde{p}_1 \end{bmatrix}, \begin{bmatrix} v_S & -u_S \end{bmatrix}, d_1, d_2 \right) \begin{bmatrix} p_2 & q_2 \end{bmatrix}$$

end if

end procedure

Algorithm 8 Algorithm to calculate the uniform samples of a signal $x[k]$, $k = 0, \dots, n - 1$ from the non-uniform samples $\tilde{x}[k]$ corresponding to locations σ_i by applying the pseudoinverse of a resampling matrix with an SSD.

procedure $x = \text{NONUNI TO UNI RESAMPLE}(\tilde{x}, \sigma, N)$

$$c_i \leftarrow \pi \sigma_i$$

$$d_j \leftarrow \pi j \quad (j = 0, \dots, N - 1)$$

$$z_i \leftarrow \sin(c_i)$$

► Find generators, main diagonal of G

$$\alpha_i \leftarrow -z_i^2$$

$$u \leftarrow \text{FMM}(\alpha, d, c)$$

$$\beta \leftarrow \text{FMM}(v, c, d)$$

$$\beta_j \leftarrow -\sin(\pi \sigma_j) \cdot \beta_j$$

$$\lambda \leftarrow \text{FMM}(\beta, d, c)$$

$$\lambda \leftarrow \lambda - \nabla^\dagger(u, v, d)\mathbf{1}$$

$$\begin{bmatrix} \chi & \Upsilon \end{bmatrix} \leftarrow \text{RESAMP PSEUDO INV GEN}(z, y, c, d)$$

► Solve for generators of R^\dagger

$$\begin{bmatrix} \alpha_1 & \alpha_2 \end{bmatrix} \leftarrow \text{diag}(\tilde{x}) \Upsilon$$

► Apply pseudoinverse

$$\beta_1 \leftarrow \text{FMM}(\alpha_1, d, c)$$

$$\beta_2 \leftarrow \text{FMM}(\alpha_2, d, c)$$

$$x_j \leftarrow \chi_{j1}(\beta_1)_j + \chi_{j2}(\beta_2)_j$$

end procedure

s non-zero samples) to reduce computation. Given an s -sparse vector of uniform samples, the non-uniform samples may be directly computed as a sum of s weighted sinc functions in only $O(ms)$ operations, yielding a considerable savings if $s \ll n$.

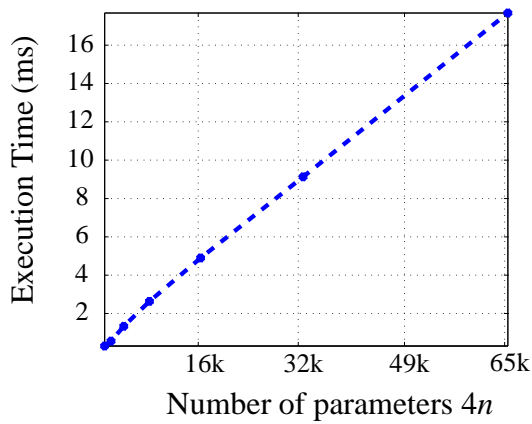
For each trial of each matrix size, a 20-sparse signal was generated as the set of true uniform samples. The non-zero components of these signals were assigned random values from a uniform distribution over $[-2, -1] \cup [1, 2]$. The true non-uniform samples were then computed by explicitly summing the 20 sampled sinc functions.

For each trial, the resampling algorithms processed the true samples and their execution times were recorded. To reduce deviations in execution time, each algorithm was run five times per trial and the recorded execution time was the average of these trials, for a total of 50000 calls to each resampling algorithm. The accuracy of the results were then compared to the directly computed or initialized values.

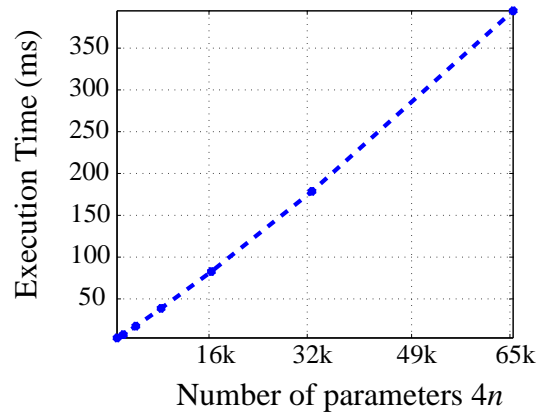
Two scenarios were considered for the non-uniform-to-uniform resampling. In the first, the generators of the pseudoinverse $G^{-1}R^T$ were computed “on the fly.” In the second, the generators were provided, and thus the only step in the non-uniform-to-uniform resampling algorithm was to apply the pseudoinverse from its SSD. The latter scenario results in significantly reduced execution time (though the accuracy is identical since both are using the same generators).

The execution time results are plotted in Figure 10. Panel (a) demonstrates that the conversion to non-uniform samples scales as expected, growing nearly linearly with the signal length as predicted by the asymptotic cost of the FMM. Panel (b) shows similar behavior for the recovery of uniform samples, with much higher overhead cost from computing the generators of the pseudoinverse. In contrast, panel (c) shows a reduced cost for recovery when the generators are known *a priori*.

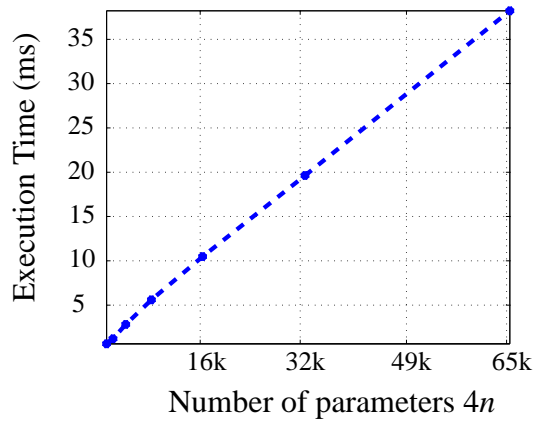
The accuracy results are plotted in Figure 11, and indicate that the resampling algorithms indeed achieve a high level of accuracy. In the figure, the distribution of the log of



(a)



(b)



(c)

Figure 10: Algorithm runtime for: (a) resampling onto a non-uniform grid, (b) recovering uniform samples from non-uniform samples, and (c) recovering uniform samples with pre-computed generators of G^{-1} .

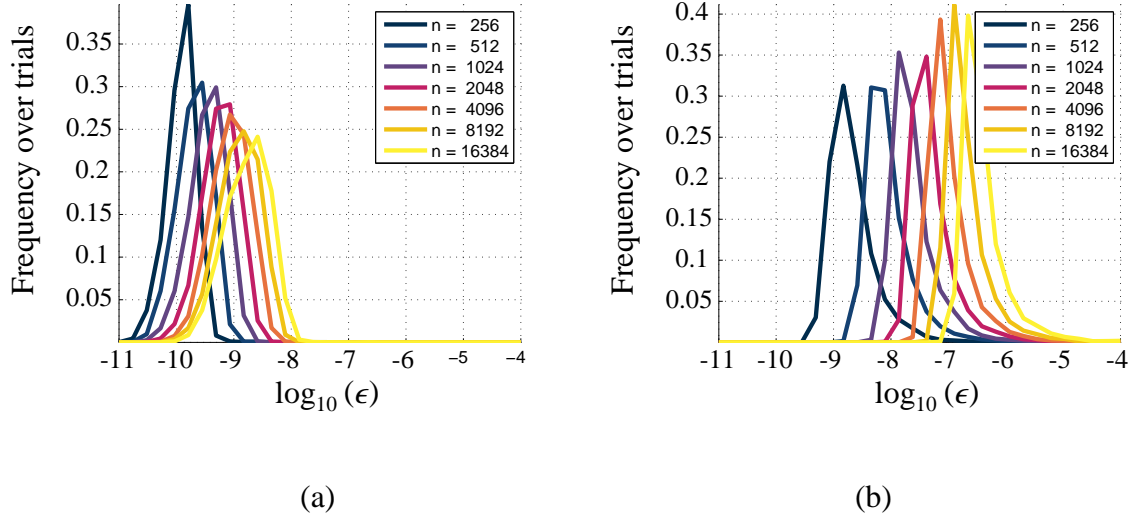


Figure 11: *Distribution of the log of the normalized errors across experimental trials for: (a) resampling onto a non-uniform grid, (b) recovering uniform samples from non-uniform samples. Each distribution was calculated by computing a histogram of the log of the normalized errors using 30 bins evenly spaced between -11 and -4 .*

the normalized errors

$$\epsilon = \frac{\|x^* - x\|_2}{\|x\|_2}.$$

across the 10000 trials for each matrix size are plotted. As the matrix size increases, more numerical error propagates and the algorithms yield fewer digits of precision in the solution.

The distribution of errors for the two problems behave differently. The distribution of errors in the uniform-to-non-uniform resampling algorithm has a larger tail to the left. This behavior is indicative of the inherent accuracy bound of the FMM. Since some minimal degree of precision is assured by the FMM, for a given matrix size one would expect a sharp drop in the distribution around some point corresponding to the “guaranteed precision” of the FMM. However, the location and amplitude of the sparse components of the input signal and the non-uniform grid points were chosen at random, and therefore any given random draw may result in more or less accurate solutions. As a result, the left side of the distribution has a longer tail.

By contrast, the FMM is only a single component of the non-uniform-to-uniform resampling algorithm. Therefore, while the FMM has some minimal precision, the conditioning of the various subproblems encountered in the divide-and-conquer approach may increase the error level. Since this algorithm is inverting the Gramian of the resampling matrix, the conditioning of the problem is sensitive to the distribution of the non-uniform sampling nodes. If a region of the uniform grid is underrepresented in the non-uniform grid, it will lead to a less well-conditioned subproblem to solve when determining the generators of the pseudoinverse. As a result of this behavior, the error distributions for the non-uniform-to-uniform resampling experiments have larger tails to the right.

It is possible that the accuracy results for the non-uniform-to-uniform resampling could be improved by introducing pivoting during the generator calculation. However, the design of a good pivoting strategy is more nuanced for these matrices. In general, pivoting will change the Toeplitz matrices that are used to apply the blocks of G into Cauchy blocks. These blocks would then be applied with several FMM calls rather than FFT calls. Since the FMM is not exact, it is unclear whether using standard pivoting techniques would provide tangible accuracy benefits. An “interleaving” pivoting strategy would preserve Toeplitz structure, but its effect on accuracy may not be significant. It is possible, however, that iterative refinement would improve the accuracy at the cost of two FMMs and four DMMs per iteration.

3.2.5 Applications of interest

The algorithms of Sections 3.2.1 and 3.2.3 prescribe algorithms for superfast non-uniform resampling. The numerical simulations of Section 3.2.4 provide evidence that the performance of these algorithms scales properly with the problem size and that the algorithms return results with a reasonable accuracy. It remains to discuss in what capacity these algorithms might function for practical signal-processing problems.

3.2.5.1 *Non-uniform resampling for non-uniform FFTs*

In Section 3.1.5.5, the Tikhonov-regularization algorithm was used to recover samples of a signal from non-uniform samples of its spectrum. Since the Gramian of the non-uniform DFT matrix A is Toeplitz, this formulation led to a Toeplitz-structured regularization problem. However, one may also consider treating the problem in an alternative way, first performing a non-uniform resampling in the Fourier domain and then transforming the interpolated coefficients back into temporal/spatial samples. This approach mirrors the classical approach to non-uniform FFTs, which take an oversampled FFT of the input signal and use the samples to interpolate the spectrum at arbitrary points [87].

In practice, the oversampled spectrum is often interpolated by a truncated kernel to reduce computation [65]. However, Algorithm 5 may be employed for the problem if a sinc kernel is used for the interpolation. The task of computing a non-uniform FFT would then reduce to a single oversampled FFT, FMM, and DMM.

To demonstrate the potential utility of superfast non-uniform resampling for NUFFT, a series of simulations were run with the following setup. For a total of 10000 experiments, a length $n = 1024$ signal was generated by drawing coefficients from a normal distribution and then windowing them with a Hann window. For every 500 trials within the 10000, a new set of $3n$ frequencies were chosen from a uniform random distribution over $[-1/2, 1/2)$ and a non-uniform DFT matrix was explicitly constructed. For each individual trial within the set, the exact non-uniform DFT was calculated by applying the non-uniform DFT matrix to the input samples. The non-uniform resampling algorithm was then used to calculate the samples, using a periodic extension to improve the accuracy of the calculated samples at the edges of the main frequency interval. The times required for direct application and for the non-uniform resampling were recorded and the relative error of the resampling process was calculated.

Once again, all experiments were run on a 3.16 GHz Intel Core 2 Duo machine with 3.0 GB of RAM under the Ubuntu 12.04 LTS operating system and MATLAB R2012A. The

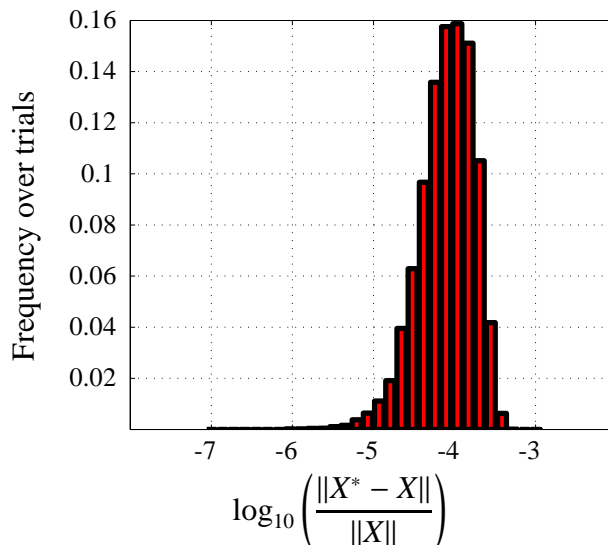


Figure 12: Distribution of the log of the normalized errors across experimental trials for the calculation of NUFFTs. The distribution was calculated by computing a histogram of the log of the normalized errors using 30 bins evenly spaced between -7 and -3 .

code implementation was the same as used for Section 3.2.4. The average time required to use non-uniform resampling to calculate the non-uniform DFT was 6.90 ms, compared to 178.2 ms for a direct calculation of the non-uniform DFT, for a speedup factor of 25.8. The distribution of errors in the calculations is shown in Figure 12. While the relative errors are larger than for the performance scaling simulations, the non-uniform resampling algorithm yields reasonable results with much greater speed than direct computation.

3.2.5.2 Converting level crossings to uniform samples

In standard sampling architectures, a sample is acquired by taking a “snapshot” of a signal at a specified point in time and the output is quantized and recorded. For high sampling frequencies, the process of time synchronization and quantization can significantly complicate the hardware design. It would therefore be beneficial to use alternative sampling architectures that might reduce the hardware burden for high-frequency sampling.

One option is to use a level-crossing architecture. In this configuration, a set of specified “levels” each have a sensor that records the time that a signal’s value crosses that level. Since each sensor covers its own level, the sample acquisition process amounts to recording

only timing information. Since the input signal will not, in general, cross levels at even time steps, the resulting samples are non-uniformly spaced in time.

The appeal of a level-crossing architecture is that the hardware design is significantly simpler. However, there is a practical need for efficient algorithms to resample the acquired data from these architectures onto uniform grids. If the sinc kernel provides a good interpolation kernel, then Algorithm 8 would have potential benefit for this application.

CHAPTER IV: MULTI-LEVEL TOEPLITZ THEORY AND ALGORITHMS

4.1 *Inversion for special classes*

In general, the superfast algorithms that have been developed for Toeplitz systems do not extend well to two-level Toeplitz matrices. However, when either level of a two-level Toeplitz matrix contains a commutative structure, the inverse can be expressed as an SSD from a small number of generators. As a result, standard scalar algorithms for computing the inverse generators can be extended to these matrices, and systems involving this structure can be solved with a few 2-D FFTs [103].

4.1.1 Commutative classes of Toeplitz matrices

Two subclasses of Toeplitz matrices have the property of commutativity, and thus form commutative rings. A commutative ring \mathcal{R} consists of a set of elements \mathcal{S} and associated operations for addition (+) and multiplication (*), and is denoted $\mathcal{R} = (\mathcal{S}, +, *)$. For \mathcal{R} to be a commutative ring, its elements must satisfy the following properties [42]:

1. $A + X = B$ is solvable for all X (existence of an additive inverse);
2. $A + (B + C) = (A + B) + C$ (associativity of addition);
3. $A(BC) = (AB)C$ (associativity of multiplication);
4. $A(B + C) = AB + AC$ and $(B + C)A = BA + CA$ (distributivity of multiplication);
5. $A + B = B + A$ (commutativity of addition); and
6. $AB = BA$ (commutativity of multiplication).

The properties of matrix addition and multiplication assure that any generic set of $n \times n$ matrices satisfies all but the last criteria. The final criterion implies a special and highly-restrictive property: any commutative ring of matrices is closed under inversion (for any

nonsingular elements). That is, given a commutative ring \mathcal{R} of square matrices, the inverse of any nonsingular element in \mathcal{R} must also be in \mathcal{R} .

There are two particular subclasses of Toeplitz matrices that form commutative rings.

Lemma 4.1. *Let \mathcal{S} be the set of all $n \times n$ upper or lower-triangular Toeplitz matrices and define the binary operations $+$ and $*$ to be matrix addition and multiplication, respectively. Then $(\mathcal{S}, +, *)$ is a commutative ring.*

Proof. See [46], Proposition 10.1.1. □

Lemma 4.2. *For any $f \in \mathbb{Z}$ such that $|f| = 1$, let \mathcal{S} be the set of all $n \times n$ f -circulant matrices and define the binary operations $+$ and $*$ to be matrix addition and multiplication, respectively. Then $(\mathcal{S}, +, *)$ is a commutative ring.*

Proof. Since any element $A \in \mathcal{S}$ is f -circulant, it can be written as

$$A = \sum_{i=0}^{n-1} Z_f^i a_i$$

for $a_i \in \mathbb{C}$. Since $|f| = 1$, let f be written as $f = \mathbf{e}^{j\theta}$ for some $\theta \in (-\pi, \pi]$. Define an $n \times n$ diagonal matrix $\Lambda = \text{diag}([\mathbf{e}^{j\theta i/n}])$; then

$$\Lambda Z_f \bar{\Lambda} = \begin{bmatrix} 0 & \cdots & 0 & \mathbf{e}^{j\theta} \\ \mathbf{e}^{j\theta/n} & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \\ 0 & \cdots & \mathbf{e}^{j\theta(n-1)/n} & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \mathbf{e}^{-j\theta} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{e}^{-j\theta(n-1)/n} \end{bmatrix} = \mathbf{e}^{j\theta/n} Z_1.$$

The matrix Z_1 is circulant, and therefore $Z_1 = \mathbf{F}^H \text{diag}([\mathbf{e}^{-j2\pi k/n}]) \mathbf{F} = \mathbf{F}^H \Sigma \mathbf{F}$. Therefore,

$$Z_f = \bar{\Lambda} \mathbf{F}^H (\mathbf{e}^{j\theta/n} \Sigma) \mathbf{F} \Lambda.$$

Since $|f| = 1$,

$$\begin{aligned} Z_f^2 &= Z_f Z_f = \bar{\Lambda} \mathbf{F}^H (\mathbf{e}^{j\theta/n} \Sigma) \mathbf{F} \Lambda \bar{\Lambda} \mathbf{F}^H (\mathbf{e}^{j\theta/n} \Sigma) \mathbf{F} \Lambda \\ &= \bar{\Lambda} \mathbf{F}^H (\mathbf{e}^{j2\theta/n} \Sigma^2) \mathbf{F} \Lambda, \end{aligned}$$

and more generally $Z_f^i = \bar{\Lambda} \mathbf{F}^H \left(\mathbf{e}^{i\theta/n \Sigma^i} \right) \mathbf{F} \Lambda$. Therefore, any $A \in \mathcal{S}$ can be written as

$$A = \bar{\Lambda} \mathbf{F}^H \left(\sum_{i=0}^{n-1} \mathbf{e}^{i\theta/n \Sigma^i} a_i \right) \mathbf{F} \Lambda = \bar{\Lambda} \mathbf{F}^H D_A \mathbf{F} \Lambda,$$

where D_A is diagonal. For two matrices $A, B \in \mathcal{S}$,

$$\begin{aligned} AB &= \bar{\Lambda} \mathbf{F}^H D_A \mathbf{F} \Lambda \bar{\Lambda} \mathbf{F}^H D_B \mathbf{F} \Lambda = \bar{\Lambda} \mathbf{F}^H D_A D_B \mathbf{F} \Lambda \\ &= \bar{\Lambda} \mathbf{F}^H D_B D_A \mathbf{F} \Lambda = \bar{\Lambda} \mathbf{F}^H D_B \mathbf{F} \Lambda \bar{\Lambda} \mathbf{F}^H D_A \mathbf{F} \Lambda = BA. \end{aligned}$$

Therefore, all elements $A, B \in \mathcal{S}$ commute. □

The set described in Lemma 4.2 has another interesting property, as well: it contains nearly all normal Toeplitz matrices [34].

4.1.2 Inversion formulas and algorithms

Section 2.1.1 gave the Gohberg-Semencul SSDs for scalar Toeplitz inverses, which contain only triangular Toeplitz matrices. When a two-level Toeplitz matrix contains circulant or triangular structure in one or more of its levels, these formulas can be extended naturally through the use of Kronecker products. The term **semi-commutative two-level Toeplitz** (SCTLT) will be used to refer to this class of matrices for brevity. The Gohberg-Semencul formulas allow for efficient inversion of SCTLT matrices.

4.1.2.1 Gohberg-Semencul formulas for SCTLT matrices

It is possible to follow the exact progression of Section 2.1 to obtain a broad and generalized form of the Gohberg-Semencul formulas for SCTLT matrices. However, given that the formulations of 2.1.1.3 follow immediately from a few simple results, it is easiest to instead define a single set of generators and a specific Gohberg-Semencul formula. The generalization to equivalent generator sets and formulas may then be obtained in a fashion similar to that of Section 2.1.

In Section 1.5.1, the level-swapping matrix was used to exchange the levels of a two-level matrix. Without loss of generality, then, one may consider a two-level Toeplitz matrix

T whose blocks A_{i-j} are members of a commutative ring of Toeplitz matrices. For matrices with the opposite level structure, all results and algorithms hold after the matrix is permuted on both sides with the level-swapping matrix.

Scalar Toeplitz displacement expressions can be extended to the block level through the use of Kronecker products. Rather than the f -shift matrix Z_f , a convenient block-level displacement uses the block-level f -shift matrix X_f .

Definition 4.1 (Block-level f -shift matrix). *The (m, n) block-level f -shift matrix X_f is defined as $X_f = Z_f \otimes I_m$.*

Similarly structured – but non-separable – matrices may instead be used, but X_f serves as a particularly convenient choice. Much as for scalar matrices, the block-level f -shift matrix can be used to define circulant-like structure.

Definition 4.2 (Block-level f -circulant matrix). *A matrix $A \in \mathbb{C}^{mn \times mn}$ is block-level f -circulant if there are matrices $A_i \in \mathbb{C}^{m \times m}$ such that*

$$A = \sum_{i=0}^{n-1} X_f^i (I_n \otimes A_i) = \begin{bmatrix} A_0 & fA_{n-1} & \cdots & fA_1 \\ A_1 & A_0 & \cdots & fA_2 \\ \vdots & \vdots & \ddots & \vdots \\ A_{n-1} & A_{n-2} & \cdots & A_0 \end{bmatrix}. \quad (56)$$

Using the block-level 0-shift matrix, the displacement $\nabla_{X_0, X_0}(\cdot)$ of an SCTLT matrix has a closed-form expression similar to (17).

Proposition 4.1. *Let $T = [A_{i-j}]$ be (m, n) two-level Toeplitz with Toeplitz blocks $A_{i-j} \in \mathcal{R}$, where \mathcal{R} is a ring of commutative Toeplitz matrices. Then*

$$\nabla_{X_0, X_0}(T) = \mathcal{H}\mathcal{E}_1^P - \mathcal{E}_1\mathcal{H}^P, \quad (57)$$

where $\mathcal{E}_k = e_k \otimes I_m$ and \mathcal{H} is the block column vector

$$\mathcal{H} = \begin{bmatrix} \mathbf{0} & A_{1-n} & \cdots & A_{-1} \end{bmatrix}^T. \quad (58)$$

Proof. Using the definition of the displacement,

$$\begin{aligned}
\nabla_{X_0, X_f}(T) &= X_0 T - T X_0 \\
&= \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ A_0 & \cdots & A_{2-n} & A_{1-n} \\ \vdots & \ddots & \vdots & \vdots \\ A_{n-2} & \cdots & A_0 & A_{-1} \end{bmatrix} - \begin{bmatrix} A_{-1} & \cdots & A_{1-n} & \mathbf{0} \\ A_0 & \cdots & A_{2-n} & \mathbf{0} \\ \vdots & \ddots & \vdots & \vdots \\ A_{n-2} & \cdots & A_0 & \mathbf{0} \end{bmatrix} \\
&= \mathcal{H} \mathcal{E}_n^\mathcal{T} - \mathcal{E}_1 (\mathcal{J}_{m,n} \mathcal{H})^\mathcal{T} \\
&= \mathcal{H} \mathcal{E}_1^p - \mathcal{E}_1 \mathcal{H}^p = \mathcal{H} \mathcal{E}_1^p - \mathcal{E}_1 \mathcal{H}^p.
\end{aligned}$$

□

The block-level Sylvester displacement of the inverse of a nonsingular SCTLT matrix follows immediately.

Proposition 4.2. *For T a nonsingular (m, n) SCTLT matrix with blocks $A_{i-j} \in \mathcal{R}$, where \mathcal{R} is a ring of commutative Toeplitz matrices, let $B = T^{-1}$ with blocks $B_{i,j} \in \mathbb{C}^{m \times m}$. Define $\mathcal{U} = B \mathcal{E}_1$ and $\mathcal{V} = B \mathcal{H}$, with \mathcal{H} as in (58). Then*

$$\nabla_{X_0, X_0}(B) = \mathcal{U} \mathcal{V}^p - \mathcal{V} \mathcal{U}^p. \quad (59)$$

Proof. From the properties of Sylvester displacement,

$$\nabla_{X_0, X_0}(B) = -B \nabla_{X_0, X_0}(T) B = B \mathcal{E}_1 \mathcal{H}^p B - B \mathcal{H} \mathcal{E}_1^p B = \mathcal{U} \mathcal{H}^p B - \mathcal{V} \mathcal{E}_1^p B.$$

Since T is persymmetric B is as well, and from Proposition 1.5 $Q^p B = (BQ)^p$ for any $Q \in \mathbb{C}^{mn \times m}$. Therefore,

$$\nabla_{X_0, X_0}(B) = \mathcal{U} \mathcal{V}^p - \mathcal{V} \mathcal{U}^p.$$

□

The operator $\nabla_{X_0, X_0}(\cdot)$ has a nullspace equal to the set of block-lower-triangular two-level Toeplitz matrices.

Proposition 4.3. *Let $A \in \mathbb{C}^{mn \times mn}$; then $\nabla_{X_0, X_0}(A) = \mathbf{0}$ if and only if A is an (m, n) two-level matrix with a lower-triangular Toeplitz block pattern; that is, if A is of the form*

$$A = \begin{bmatrix} A_0 & \mathbf{0} & \cdots & \mathbf{0} \\ A_1 & A_0 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n-1} & A_{n-2} & \cdots & A_0 \end{bmatrix}. \quad (60)$$

for some matrices $A_i \in \mathbb{C}^{m \times m}$.

Proof. The proof follows the form of the proof of Proposition 2.5, with the scalars $a_{i,j} = a_{i-j}$ replaced with the matrices $A_{i,j} = A_{i-j}$. Since all operations involved require only matrix additions, the blocks are not required to be commutative. \square

Using Proposition 4.3, a Gohberg-Semencul formula for SCTLT matrices can be stated.

Theorem 4.1. *For T a nonsingular (m, n) SCTLT matrix with blocks $A_{i-j} \in \mathcal{R}$, where \mathcal{R} is a ring of commutative Toeplitz matrices, let $B = T^{-1}$ with blocks $B_{i,j} \in \mathbb{C}^{m \times m}$. Define $\mathcal{U} = B\mathcal{E}_1$ and $\mathcal{V} = B\mathcal{H}$, with \mathcal{H} as in (58). Then the blocks $\mathcal{U}_i, \mathcal{V}_i \in \mathcal{R}$ and the matrix B can be decomposed as*

$$B = \left(\mathbf{I} - \begin{bmatrix} \mathbf{0} & \mathcal{V}_n & \cdots & \mathcal{V}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathcal{V}_n \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix} \right) \begin{bmatrix} \mathcal{U}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathcal{U}_2 & \mathcal{U}_1 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{U}_n & \mathcal{U}_{n-1} & \cdots & \mathcal{U}_1 \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathcal{U}_n & \cdots & \mathcal{U}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathcal{U}_n \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathcal{V}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathcal{V}_2 & \mathcal{V}_1 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{V}_n & \mathcal{V}_{n-1} & \cdots & \mathcal{V}_1 \end{bmatrix} \quad (61)$$

Proof. See Appendix A. \square

The Gohberg-Semencul formula in Theorem 4.1 is an extension of the Barnett formula [8]. Since the components of \mathcal{U} and \mathcal{V} are from the ring \mathcal{R} , it is an SSD of the SCTLT inverse that uses only block-triangular SCTLT matrices.

Moreover, the elements of both commutative Toeplitz rings are parametrized by n coefficients. Therefore, the block columns \mathcal{U} and \mathcal{V} are entirely described by a total of mn

parameters each. For both the f -circulant and triangular rings, the generators \mathcal{U} and \mathcal{V} are in fact entirely described by a single column.¹ The generators \mathcal{U} and \mathcal{V} of the matrix B can then be obtained by solving systems of the form $B\mu = \nu$.

4.1.2.2 Inverse generator calculation for the triangular Toeplitz ring

One method of calculating the inverse generators of SCTLT matrices with triangular blocks is to generalize the tangential-interpolation algorithm, replacing scalars with triangular Toeplitz matrices. However, this is a poor strategy in practice; as the block size increases the interpolation becomes unstable as a result of the notorious ill-conditioning of triangular Toeplitz matrices. A more attractive alternative is to extend the superfast triangular-Toeplitz inversion algorithm of [24] to the block-Toeplitz case. This algorithm recursively computes the inverse generator of a principal leading submatrix (PLS) and then uses it to solve for the inverse generator of the next largest PLS.

Without loss of generality, as the results extend easily to upper-triangular Toeplitz matrices, let \mathcal{R} be the commutative ring of lower-triangular Toeplitz matrices. Denote the set of SCTLT matrices whose blocks are elements of \mathcal{R} as $\mathbb{T}[\mathcal{R}]$. For a nonsingular matrix $T \in \mathbb{T}[\mathcal{R}]$, given the components \mathcal{U}, \mathcal{V} it is clear from the SSD in Theorem 4.1 how to apply the matrix in $O(mn \log mn)$ operations with 2-D FFTs. However, as is the case with most algorithms based on inverse characterization, solving for these components is costly if generic algorithms are used.

Let P be the (m, n) level-swapping matrix, and define $\tilde{T} = P^T T P$. Then \tilde{T} is block lower-triangular Toeplitz with Toeplitz blocks, and solving a system of the form $T\mu = \nu$ is equivalent to computing $\mu = P\tilde{T}^{-1}P^T\nu$. Assuming m to be even, \tilde{T} can be partitioned as

$$\tilde{T} = \begin{bmatrix} \tilde{T}_0 & \mathbf{0} \\ \tilde{T}_1 & \tilde{T}_0 \end{bmatrix},$$

¹Circulants and lower-triangular Toeplitz matrices are entirely described by their first column, upper-triangular Toeplitz matrices by their last column.

where $\widetilde{T}_0, \widetilde{T}_1 \in \mathbb{C}^{mn/2 \times mn/2}$. Moreover, the matrix \widetilde{T}_0 is also block lower-triangular Toeplitz with Toeplitz blocks.

From the block LDU decomposition of Section 2.3, the partitioned inverse of \widetilde{T} is

$$\widetilde{T}^{-1} = \begin{bmatrix} \widetilde{T}_0^{-1} & \mathbf{0} \\ -\widetilde{T}_0^{-1}\widetilde{T}_1\widetilde{T}_0^{-1} & \widetilde{T}_0^{-1} \end{bmatrix}.$$

Letting $\widetilde{v} = P^T v$ and $\widetilde{\mu} = P^T \mu$, the permuted solution is

$$\begin{bmatrix} \widetilde{\mu}_1 \\ \widetilde{\mu}_2 \end{bmatrix} = \begin{bmatrix} \widetilde{T}_0^{-1} & \mathbf{0} \\ -\widetilde{T}_0^{-1}\widetilde{T}_1\widetilde{T}_0^{-1} & \widetilde{T}_0^{-1} \end{bmatrix} \begin{bmatrix} \widetilde{v}_1 \\ \widetilde{v}_2 \end{bmatrix}.$$

Its components can thus be written as

$$\widetilde{\mu}_1 = \widetilde{T}_0^{-1}\widetilde{v}_1 \quad \text{and} \quad \widetilde{\mu}_2 = \widetilde{T}_0^{-1}(\widetilde{v}_2 - \widetilde{T}_1\widetilde{\mu}_1).$$

When solving the system $T\mu = v$ for generic vectors, this expression implies that the problem can be subdivided. First, a system one quarter the size may be solved to determine $\widetilde{\mu}_1$, and then the same system may be solved for a new input $(\widetilde{v}_2 - \widetilde{T}_1\widetilde{\mu}_1)$. Since \widetilde{T}_1 is $(m/2, n)$ two-level Toeplitz, the updated input can be computed efficiently with 2-D FFTs. The recursion is stopped when \widetilde{T}_0 is reduced to a scalar Toeplitz matrix, at which point any scalar Toeplitz inversion method may be used.

For the particular case when μ are the generators of T , the computation can be made even more efficient by using the fact that the columns of $\widetilde{\mu}_1$ are actually the inverse generators of \widetilde{T}_0 . Therefore, once $\widetilde{\mu}_1$ is computed, the SSD of the matrix \widetilde{T}_0^{-1} in Theorem 4.1 can be used to solve the second system. Since all operations to apply \widetilde{T}_0^{-1} and \widetilde{T}_1 involve 2-D FFTs, this results in the $\mathcal{O}(mn \log^2 mn)$ procedure given in Algorithm 9.

4.1.2.3 Inverse generator calculation for the circulant Toeplitz ring

The task of inverting SCTLT matrices whose blocks are f -circulant is simpler than the triangular case. For simplicity, let \mathcal{R} be the ring of $m \times m$ circulant matrices. The same

Algorithm 9 Algorithm to compute the inverse generators u and v of an (m, n) SCLTL matrix with triangular Toeplitz blocks.

procedure $[u \ v] = \text{SCLTLINVGENTRI}(T, m, n, \text{type})$

if type is 'lower' **then**

$$\begin{bmatrix} g & h \end{bmatrix} \leftarrow \begin{bmatrix} e_1 & X_0 T e_{mn-m+1} \end{bmatrix}$$

else

$$\begin{bmatrix} g & h \end{bmatrix} \leftarrow \begin{bmatrix} e_m & X_0 T e_{mn} \end{bmatrix}$$

end if

if $m = 1$ **then**

$$\begin{bmatrix} u & v \end{bmatrix} \leftarrow T^{-1} \begin{bmatrix} g & h \end{bmatrix}$$

else

$$\begin{bmatrix} u_1 & v_1 \end{bmatrix} = \text{SCLTLINVGENTRI}(\widetilde{P}T_0P^T, m/2, n, \text{type})$$

$$\begin{bmatrix} u_2 & v_2 \end{bmatrix} \leftarrow \widetilde{P}T_0^{-1} \left(P^T \begin{bmatrix} g_2 & h_2 \end{bmatrix} - \widetilde{T}_1 P^T \begin{bmatrix} u_1 & v_1 \end{bmatrix} \right)$$

end if

end procedure

results apply for f -circulant rings if the DFT matrix is replaced with the appropriate unitary matrix.

Let the SCLTL matrices whose blocks are in \mathcal{R} again be denoted as $\mathbb{T}[\mathcal{R}]$. Suppose P is the (m, n) level-swapping matrix; then $\widetilde{T} = P^T T P$ has a circulant block pattern and its blocks are Toeplitz matrices. Since the block pattern is circulant, the matrix \widetilde{T} can then be factored as

$$\widetilde{T} = (\mathbf{F}^H \otimes I_n) D (\mathbf{F} \otimes I_n),$$

where D is a block diagonal matrix whose blocks D_{ii} are

$$D_{i,i} = \sum_{k=0}^{m-1} \widetilde{T}_{k,1} \mathbf{e}^{-j2\pi ik/m}.$$

This decomposition implies

$$\widetilde{T}^{-1} = (\mathbf{F}^H \otimes I_n) D^{-1} (\mathbf{F} \otimes I_n),$$

and since D is block diagonal the computation of D^{-1} involves m separate $n \times n$ scalar Toeplitz inversions. If the scalar Toeplitz inversions are performed in $O(n \log^2 n)$ operations

with a superfast solver, then the asymptotic cost of computing the inverse generators of \widetilde{T} is $O(mn \log^2(\max(m, n)))$.

Algorithm 10 Algorithm to compute the inverse generators u and v of an (m, n) SCTLT matrix with circulant blocks.

procedure $[u \ v] = \text{SCTLTINVGENCIRC}(T, m, n)$

$$\begin{bmatrix} g & h \end{bmatrix} \leftarrow \begin{bmatrix} e_1 & X_0 T e_{mn-m+1} \end{bmatrix}$$

$$\begin{bmatrix} \check{g} & \check{h} \end{bmatrix} \leftarrow (\mathbf{F} \otimes I_n) P^T \begin{bmatrix} g & h \end{bmatrix}$$

for $i = 0, \dots, m - 1$ **do**

$$D_{i,i} \leftarrow \sum_{k=0}^{m-1} \widetilde{T}_{k,i} \mathbf{e}^{-j2\pi ik/m}$$

$$\begin{bmatrix} \check{u}_i & \check{v}_i \end{bmatrix} \leftarrow D_{i,i}^{-1} \begin{bmatrix} \check{g}_i & \check{h}_i \end{bmatrix}$$

end for

$$\begin{bmatrix} u & v \end{bmatrix} \leftarrow P(\mathbf{F}^H \otimes I_n) \begin{bmatrix} \check{u} & \check{v} \end{bmatrix}$$

end procedure

These steps are given in Algorithm 10, which computes the generators of an SCTLT matrix with circulant blocks. The only steps required to adapt the algorithm for f -circulant blocks are the additions of DMMs in the unitary Fourier transforms. Otherwise, the algorithm functions in exactly the same manner.

4.1.3 Numerical results

The accuracy and speed of the superfast inversion algorithm of Section 4.1.2.2 is demonstrated through numerical simulations. First, to show the accuracy of the inversion algorithm in a practical setting, it was used to deblur nine natural images of varying sizes that were corrupted with blurring filters.² Each blurring filter was the sum of a windowed bivariate Gaussian function with random correlation coefficient ρ and a weighted impulse. The weighted impulse was used to improve the matrix conditioning and effectively serves as a form of regularization.

²The natural images were selected from a data set provided by Alex Teitelbaum.

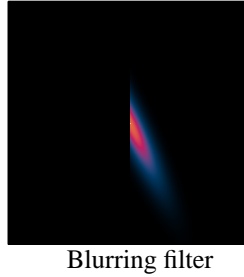


Figure 13: Example blurring filter used in the recovery experiments.

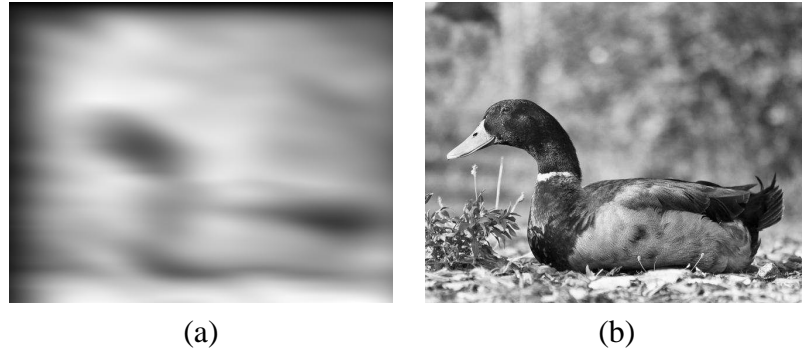


Figure 14: Images generated from the deblurring experiment: (a) an example blurred image and (b) the recovered image.

The impulse response of these blurring filters took the form

$$h[i, j] = \alpha\delta[i, j] + g[i, j]u[j - n],$$

where $u[j]$ is the unit step and $g[i, j]$ is the bivariate Gaussian. An example of such a filter is displayed in Figure 13. The unit step in the impulse response was used to assure that the convolution matrix of the filter had triangularity in one of its levels, and the convolution was thus the result of an SCTLT matrix-vector multiplication.

After corrupting the input images with the blurring filter, the inverse generators for the convolution matrix were computed using Algorithm 9. Using the generators, the inverse convolution matrix was then applied to each corrupted image through its SSD to produce a restored version. The pSNR of the restored images were then computed using the ground truth images.

Iterative refinement as described in [111] was used to improve the recovery estimate, as its implementation in this case is straightforward. Moreover, since the conditioning of

Table 8: Accuracy and execution times vs. image size for the deblurring experiment. The number of entries in the blurring matrix is the square of the number of pixels in the image. The fifth column is the time required to apply three stages of iterative refinement to the solution, each of which requires an application of both the matrix and the inverse.

Num. Pixels	Blur Time (s)	CFS Sol. Time (s)	Inv. Time (s)	Iter. Ref. Time (s)	PSNR (dB)
273K	0.497	4.18	1.03	3.98	202.7
336K	0.787	4.76	1.45	5.46	198.0
396K	0.987	4.78	1.55	5.80	201.8
546K	1.08	9.08	2.44	10.13	177.6
593K	1.35	9.29	2.60	9.29	205.4
699K	1.98	9.34	2.68	10.5	185.0
927K	2.09	12.2	3.49	12.9	143.2
1.66M	3.71	21.0	7.65	26.7	193.9
2.29M	5.59	37.6	13.1	42.3	201.6

triangular Toeplitz matrices can be problematic, iterative refinement can dramatically increase the accuracy of the restored image at the cost of a few matrix and inverse applications with the FFT.

Figure 14 displays examples of blurred and recovered images. Table 8 lists the pSNR and execution times for the various images. All images are recovered rapidly and to a high degree of accuracy (pSNR above 100 dB implies image recovery to within quantization levels).

A second set of experiments was used to verify that the algorithm’s performance scaled properly with the problem size. In these experiments, synthetic phantom images were generated and blurred with the same types of filters as before. For the first portion of these experiments, the block size m was varied while the number of blocks n was held constant at 256. The execution times required to calculate the inverse generators and to apply the matrix T^{-1} are plotted as a function of the block size in Figure 15.

For the second portion, the block size was held constant at 256 while the number of blocks was varied. The execution times are again plotted as a function of the number of blocks in Figure 16. For each portion, the matrix size grew linearly as the variable parameter is changed. No iterative refinement was applied in either portion of the experiment. As

can be seen in Figures 15 and 16, the time required to calculate the inverse parameters and to apply the inverse scales nearly linearly with the matrix size.

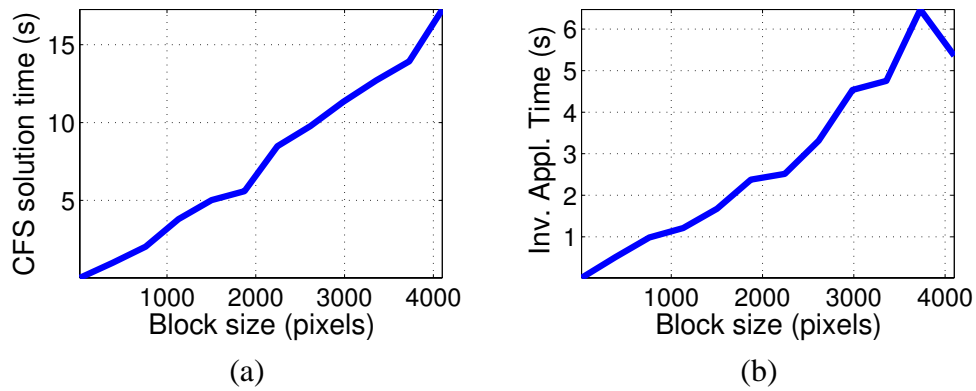


Figure 15: Execution times for (a) calculation of the CFS and (b) application of the inverse blurring matrix to the input vector as a function of the block size. The number of blocks is held constant at 256.

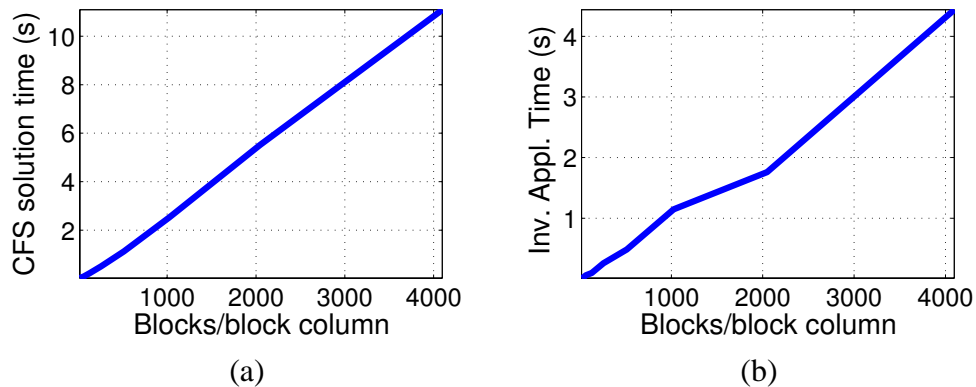


Figure 16: Execution times for (a) calculation of the CFS of the blurring matrix and (b) application of the inverse blurring matrix to the input vector as a function of the number of blocks. The block size is held constant at 256.

Algorithm 10 is straightforward to implement. The Fourier matrices applied to factorize this class of SCTLT matrices have a condition number of 1, so the accuracy of the algorithm is, from a practical point, entirely dependent on the accuracy of the inversion of the block diagonal matrix. Therefore, the performance scaling and accuracy of this algorithm are essentially the same as the chosen Toeplitz inversion routine used for inverting the central block diagonal matrix in the factorization.

4.2 *One-level formulas for two-level Toeplitz inverses*

In Section 4.1, scalar Toeplitz inversion algorithms were extended to special classes of two-level Toeplitz matrices. The principal difficulty in further extending these algorithms to generic two-level Toeplitz matrices is that the structure of the two-level Toeplitz inverse has proven elusive to identify. To date, the scalar Toeplitz inverse SSD of Gohberg and Semencul has been extended to at most one structural level of multi-level Toeplitz matrices at a time.

The most well-known example is the Gohberg-Heinig formula [43], which gives a description similar to the Gohberg-Semencul formulas for block Toeplitz matrices. The Gohberg-Heinig formula has since been used as a building block for different and equivalent block Toeplitz inverse formulas [9, 31, 32, 67, 69, 71], generalized inverses of block Toeplitz matrices [1], and extensions of other known scalar Toeplitz results [62]. By exchanging matrix levels, it is not difficult to see that all of these results have analogs for Toeplitz-block matrices.

In fact, all previously derived Gohberg-Heinig formulas for two-level Toeplitz matrices, and infinitely many more, may be obtained using the approach of Section 2.1.1.3. In short, this result may be summarized as follows:

1. there are specific pairs of block column generators that completely describe the two-level Toeplitz inverse;
2. any set of block column vectors with the same span as a pair of generators yields a Gohberg-Heinig formula for the two-level Toeplitz inverse; and
3. the matrices in *any* valid Gohberg-Heinig formula are a linear combination of *any* valid generator pair.

More simply put, the latter two points state that any valid pair of generators yields a Gohberg-Heinig formula and that all Gohberg-Heining formulas can be written in terms of any valid pair of generators [105].

4.2.1 Block-level displacement

Once again, the key to producing useful SSDs is to use displacement. Using the block-level f -shift matrices of Section 4.1.2.1, the displacements $\nabla_{X_{f_1}, X_{f_2}}(\cdot)$ and $\nabla_{X_{f_1}^T, X_{f_2}^T}(\cdot)$ of a generic two-level Toeplitz matrix yield closed-form expressions. The following are more general analogs of Propositions 4.1 and 4.2 for two-level Toeplitz matrices that do not necessarily have commutative structure.

Proposition 4.4. *Let $T = [A_{i-j}]$ be (m, n) two-level Toeplitz with Toeplitz blocks A_{i-j} . Let $R \in \mathbb{C}^{m \times m}$ be persymmetric; then*

$$\nabla_{X_0, X_f}(T) = (\mathcal{H}_{[R]} - fT\mathcal{E}_1)\mathcal{E}_1^P - \mathcal{E}_1\mathcal{H}_{[R]}^P \quad \text{and} \quad (62)$$

$$\nabla_{X_0^T, X_f^T}(T) = (\mathcal{G}_{[R]} - fT\mathcal{E}_n)\mathcal{E}_n^P - \mathcal{E}_n\mathcal{H}_{[R]}^P, \quad (63)$$

where $\mathcal{E}_k = e_k \otimes I_m$ and $\mathcal{H}_{[R]}$ and $\mathcal{G}_{[R]}$ are the block column vectors

$$\mathcal{H}_{[R]} = \begin{bmatrix} R & A_{1-n} & \cdots & A_{-1} \end{bmatrix}^T \quad \text{and} \quad (64)$$

$$\mathcal{G}_{[R]} = \begin{bmatrix} A_1 & \cdots & A_{n-1} & R \end{bmatrix}^T. \quad (65)$$

Proof. The first displacement is shown by:

$$\begin{aligned} \nabla_{X_0, X_f}(T) &= X_0T - TX_f \\ &= \begin{bmatrix} 0 & \cdots & 0 & 0 \\ A_0 & \cdots & A_{2-n} & A_{1-n} \\ \vdots & \ddots & \vdots & \vdots \\ A_{n-2} & \cdots & A_0 & A_{-1} \end{bmatrix} - \begin{bmatrix} A_{-1} & \cdots & A_{1-n} & fA_0 \\ A_0 & \cdots & A_{2-n} & fA_1 \\ \vdots & \ddots & \vdots & \vdots \\ A_{n-2} & \cdots & A_0 & fA_{n-1} \end{bmatrix} \\ &= (\mathcal{H}_{[R]} - fT\mathcal{E}_1)\mathcal{E}_n^T - \mathcal{E}_1(\mathcal{J}_{m,n}\mathcal{H}_{[R]})^T \\ &= (\mathcal{H}_{[R]} - fT\mathcal{E}_1)\mathcal{E}_1^P - \mathcal{E}_1\mathcal{H}_{[R]}^P. \end{aligned}$$

The second is shown in a similar fashion. □

Proposition 4.5. For a nonsingular (m, n) two-level Toeplitz matrix $T = [A_{i-j}]$, let $B = T^{-1}$ with blocks $B_{i,j} \in \mathbb{C}^{m \times m}$. Let $R \in \mathbb{C}^{m \times m}$ be any persymmetric matrix and define $\mathcal{U} = B\mathcal{E}_1$, $\mathcal{V}_{[R]} = B\mathcal{H}_{[R]}$, $\mathcal{W} = B\mathcal{E}_n$, and $\mathcal{X}_{[R]} = B\mathcal{G}_{[R]}$, with $\mathcal{H}_{[R]}$ and $\mathcal{G}_{[R]}$ as in (64) and (65), respectively. Then

$$\nabla_{X_0, X_0}(B) = \mathcal{U}\mathcal{V}_{[R]}^P - \mathcal{V}_{[R]}\mathcal{U}^P \quad (66)$$

$$\nabla_{X_0^T, X_0^T}(B) = \mathcal{W}\mathcal{X}_{[R]}^P - \mathcal{X}_{[R]}\mathcal{W}^P. \quad (67)$$

Proof. From the properties of Sylvester displacement,

$$\nabla_{X_0, X_0}(B) = -B\nabla_{X_0, X_0}(T)B = B\mathcal{E}_1\mathcal{H}_{[R]}^P B - B\mathcal{H}_{[R]}\mathcal{E}_1^P B = \mathcal{U}\mathcal{H}_{[R]}^P B - \mathcal{V}_{[R]}\mathcal{E}_1^P B \quad \text{and}$$

$$\nabla_{X_0^T, X_0^T}(B) = -B\nabla_{X_0^T, X_0^T}(T)B = B\mathcal{E}_n\mathcal{G}_{[R]}^P B - B\mathcal{G}_{[R]}\mathcal{E}_n^P B = \mathcal{W}\mathcal{G}_{[R]}^P B - \mathcal{X}_{[R]}\mathcal{E}_n^P B.$$

Since T is persymmetric B is as well, and from Proposition 1.5 $Q^P B = (BQ)^P$ for any $Q \in \mathbb{C}^{m \times m}$. Therefore,

$$\nabla_{X_0, X_0}(B) = \mathcal{U}\mathcal{V}_{[R]}^P - \mathcal{V}_{[R]}\mathcal{U}^P \quad \text{and}$$

$$\nabla_{X_0^T, X_0^T}(B) = \mathcal{W}\mathcal{X}_{[R]}^P - \mathcal{X}_{[R]}\mathcal{W}^P.$$

□

One may also consider a block-level Stein displacement of two-level Toeplitz matrices and their inverses.

Proposition 4.6. For an (m, n) two-level Toeplitz matrix $T = [A_{i-j}]$ and every $f \in \mathbb{C}$,

$$\Delta_{X_0, X_f^T}(T) = \mathcal{E}_1\mathcal{H}_{[0]}^P X_0^T + (T\mathcal{E}_1 - f\mathcal{H}_{[0]})\mathcal{E}_n^P \quad \text{and} \quad (68)$$

$$\Delta_{X_0^T, X_f}(T) = \mathcal{E}_n\mathcal{G}_{[0]}^P X_0 + (T\mathcal{E}_n - f\mathcal{G}_{[0]})\mathcal{E}_1^P, \quad (69)$$

where $H_{[0]}$ and $G_{[0]}$ are as given in (64) and (65), respectively.

Proof. The first displacement is shown by:

$$\begin{aligned}
\Delta_{X_0, X_f^T}(T) &= T - X_0 T X_f^T \\
&= \begin{bmatrix} A_0 & A_{-1} & \cdots & A_{1-n} \\ A_1 & A_0 & \cdots & A_{2-n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n-1} & A_{n-2} & \cdots & A_0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & \cdots & 0 \\ fA_{1-n} & A_0 & \cdots & A_{2-n} \\ \vdots & \vdots & \ddots & \vdots \\ fA_{-1} & A_{n-2} & \cdots & A_0 \end{bmatrix} \\
&= \mathcal{E}_1 H_{[0]}^P X_0^T - f H_{[0]} \mathcal{E}_n^P + T \mathcal{E}_1 \mathcal{E}_n^P,
\end{aligned}$$

and the second by similar manipulations. \square

Proposition 4.7. For a nonsingular (m, n) two-level Toeplitz matrix $T = [A_{i-j}]$, let $B = T^{-1}$. For any persymmetric $R \in \mathbb{C}^{m \times m}$, define block column vectors \mathcal{U} , $\mathcal{V}_{[R]}$, \mathcal{W} , and $\mathcal{X}_{[R]}$ as in Proposition 4.5. Then

$$\Delta_{X_0^T, X_0}(B) = \dot{\mathcal{U}} \mathcal{V}_{[R]}^P - \dot{\mathcal{V}}_{[R]} \mathcal{U}^P \quad \text{and} \quad (70)$$

$$\Delta_{X_0, X_0^T}(B) = \dot{\mathcal{W}} \mathcal{X}_{[R]}^P - \dot{\mathcal{X}}_{[R]} \mathcal{W}^P, \quad (71)$$

where $\dot{\mathcal{U}} = X_0^T \mathcal{U}$, $\dot{\mathcal{V}}_{[R]} = X_0^T \mathcal{V}_{[R]} - \mathcal{E}_n$, $\dot{\mathcal{W}} = X_0 \mathcal{W}$, and $\dot{\mathcal{X}}_R = X_0 \mathcal{X}_{[R]} - \mathcal{E}_1$.

Proof. From the definition of X_f , direct evaluation shows that $X_1^T X_1 = X_0^T X_0 + \mathcal{E}_n \mathcal{E}_1^P$ and $X_1 X_1^T = X_0 X_0^T + \mathcal{E}_1 \mathcal{E}_n^P$. Therefore, using the results of Proposition 4.5,

$$\begin{aligned}
\Delta_{X_1^T, X_0}(B) &= (X_0^T X_0 + \mathcal{E}_n \mathcal{E}_1^P) B - X_0^T B X_0 = X_0^T (X_0 B - B X_0) + \mathcal{E}_n \mathcal{U}^P \\
&= X_0^T \mathcal{U} \mathcal{V}_{[0]}^P - X_0^T \mathcal{V}_{[0]} (X_0^T - \mathcal{E}_n) = \dot{\mathcal{U}} \mathcal{V}_{[0]}^P - \dot{\mathcal{V}}_{[0]} \mathcal{U}^P.
\end{aligned}$$

Since $\mathcal{V}_{[R]} = \mathcal{V}_{[0]} + \mathcal{U}R$, it follows that

$$\begin{aligned}
\Delta_{X_0^T, X_0}(B) &= \dot{\mathcal{U}} \mathcal{V}_{[0]}^P - \dot{\mathcal{V}}_{[0]} \mathcal{U}^P = \dot{\mathcal{U}} (\mathcal{V}_{[R]} - \mathcal{U}R)^P - (X_0^T (\mathcal{V}_{[R]} - \mathcal{U}R) - \mathcal{E}_n) \mathcal{U}^P \\
&= \dot{\mathcal{U}} \mathcal{V}_{[R]}^P - \dot{\mathcal{U}} R^P \mathcal{U}^P - (X_0^T \mathcal{V}_{[R]} - \mathcal{E}_n) \mathcal{U}^P + X_0^T \mathcal{U} R \mathcal{U}^P \\
&= \dot{\mathcal{U}} \mathcal{V}_{[R]}^P - \dot{\mathcal{U}} R \mathcal{U}^P - \dot{\mathcal{V}}_{[R]} \mathcal{U}^P + \dot{\mathcal{U}} R \mathcal{U}^P = \dot{\mathcal{U}} \mathcal{V}_{[R]}^P - \dot{\mathcal{V}}_{[R]} \mathcal{U}^P.
\end{aligned}$$

The displacement $\Delta_{X_0, X_0^T}(B)$ is proven in an analogous way. \square

The block-level SSDs of the two-level Toeplitz inverse use the generator sets $\{\mathcal{U}, \mathcal{V}_{[R]}\}$ and $\{\mathcal{W}, \mathcal{X}_{[R]}\}$. These sets have the following property, which is useful when deriving a block-level analog of the Barnett formula.

Lemma 4.3. *For a nonsingular (m, n) two-level Toeplitz matrix $T = [A_{i-j}]$, let $B = T^{-1}$ with blocks $B_{i,j} \in \mathbb{C}^{m \times m}$. Define \mathcal{U} , $\mathcal{V} = \mathcal{V}_{[R]}$, \mathcal{W} , and $\mathcal{X} = \mathcal{X}_{[R]}$ as in Proposition 4.5; then*

$$\begin{bmatrix} \mathcal{U}_n & \cdots & \mathcal{U}_1 \end{bmatrix} \begin{bmatrix} \mathcal{V}_1^P & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathcal{V}_n^P & \cdots & \mathcal{V}_1^P \end{bmatrix} = \begin{bmatrix} \mathcal{V}_n & \cdots & \mathcal{V}_1 \end{bmatrix} \begin{bmatrix} \mathcal{U}_1^P & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathcal{U}_n^P & \cdots & \mathcal{U}_1^P \end{bmatrix} \quad (72)$$

$$\begin{bmatrix} \mathcal{W}_n & \cdots & \mathcal{W}_1 \end{bmatrix} \begin{bmatrix} \mathcal{X}_1^P & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathcal{X}_n^P & \cdots & \mathcal{X}_1^P \end{bmatrix} = \begin{bmatrix} \mathcal{X}_n & \cdots & \mathcal{X}_1 \end{bmatrix} \begin{bmatrix} \mathcal{W}_1^P & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathcal{W}_n^P & \cdots & \mathcal{W}_1^P \end{bmatrix} \quad (73)$$

Proof. The sum down the $-k^{\text{th}}$ diagonal of $\nabla_{X_0, X_0}(B)$ for $k = 0, \dots, n-1$ satisfies

$$S_k = \sum_{i=2}^{n-k} B_{(i-1), (i+k)} - \sum_{i=1}^{n-1-k} B_{i, (i+1+k)} = \sum_{i=1}^{n-1-k} B_{i, (i+1+k)} - B_{i, (i+1+k)} = \mathbf{0}.$$

From Proposition 4.5, the $(i, j)^{\text{th}}$ block of $\nabla_{X_0, X_0}(B)$ is

$$(\nabla_{X_0, X_0}(B))_{i,j} = \mathcal{U}_i \mathcal{V}_{n+1-j}^P - \mathcal{V}_i \mathcal{U}_{n+1-j}^P,$$

and therefore the k^{th} block of the left-hand side of (72) is

$$\begin{aligned} Q_k &= \sum_{j=1}^k \mathcal{U}_j \mathcal{V}_{k+1-j}^P - \mathcal{V}_j \mathcal{U}_{k+1-j}^P = \sum_{j=1}^k (\nabla_{X_0, X_0}(B))_{j, (n+1-j-k)} \\ &= S_{n-k} = \mathbf{0}. \end{aligned}$$

A similar procedure is used to prove (73). □

4.2.2 Singularity of block-level displacement

Since displacement is used to generate inverse formulas, it is important to know the singularity properties of the displacement operators. It was shown in Proposition 4.3 that the operator $\nabla_{X_0, X_0}(\cdot)$ has a nullspace equal to the set of block-lower-triangular two-level Toeplitz matrices. A similar result can be stated for $\nabla_{X_0^T, X_0^T}(\cdot)$.

Proposition 4.8. *Let $A \in \mathbb{C}^{mn \times mn}$; then $\nabla_{X_0^T, X_0^T}(A) = \mathbf{0}$ if and only if A is an (m, n) two-level matrix with an upper-triangular Toeplitz block pattern.*

Proof. Again, the proof follows the form of the proof of Proposition 2.5, with the scalars $a_{i,j} = a_{i-j}$ replaced with the matrices $A_{i,j} = A_{i-j}$. □

The operators $\Delta_{X_0, X_0^T}(\cdot)$ and $\Delta_{X_0^T, X_0}(\cdot)$, by contrast, are nonsingular.

Proposition 4.9. *Let $A \in \mathbb{C}^{mn \times mn}$; then*

1. $\Delta_{X_0, X_0^T}(A) = \mathbf{0}$ if and only if $A = \mathbf{0}$, and
2. $\Delta_{X_0^T, X_0}(A) = \mathbf{0}$ if and only if $A = \mathbf{0}$.

Proof. The equations in the theorem statement are equivalent to stating that the displacements are nonsingular. From [83], the eigenvalue product $\lambda_i(Z_0)\lambda_j(Z_0^T) \neq 1$ for all pairs (i, j) , and therefore $\lambda_i(X_0)\lambda_j(X_0^T) \neq 1$. The nonsingularity of the two displacements is then proven by Proposition 1.1. □

4.2.3 Inversion through block-level displacement

With the block-level displacement properties in hand, the results of the previous sections can next be used to generate Gohberg-Heinig SSDs of two-level inverses. The component matrices of the Gohberg-Heinig SSDs are block triangular-Toeplitz matrices. Since two-level Toeplitz matrices are merely more specialized block Toeplitz matrices, these formulas can be used to describe the two-level inverse. For a nonsingular two-level Toeplitz matrix T , a Gohberg-Heinig formula has either the form

$$T^{-1} = \sum_i U_i L_i \quad \text{or} \quad T^{-1} = \sum_i L_i U_i,$$

where the $\{L_i\}$ are block lower-triangular Toeplitz matrices and the $\{U_i\}$ are block upper-triangular-Toeplitz matrices. In fact, all Gohberg-Heinig formulas for persymmetric block Toeplitz matrices (including two-level Toeplitz matrices) can be written in both forms.

Proposition 4.10. *If a persymmetric matrix A can be written as $A = \sum_i Q_i R_i$ for Q_i and R_i block persymmetric, it can also be written as $A = \sum_i R_i^{\mathcal{P}} Q_i^{\mathcal{P}}$.*

Proof.

$$A = A^P = \left(\sum_i Q_i R_i \right)^P = \sum_i (Q_i R_i)^P = \sum_i R_i^P Q_i^P = \sum_i (R_i^{\mathcal{P}})^{\mathcal{P}} (Q_i^{\mathcal{P}})^{\mathcal{P}} = \sum_i R_i^{\mathcal{P}} Q_i^{\mathcal{P}}.$$

□

The following developments show that any Gohberg-Heinig formula for a two-level Toeplitz matrix T may be viewed as a reconstruction of T^{-1} from some linear combination of its block-level generators. The following lemma explains how block triangular-Toeplitz matrices arise in the recovery equations.

Lemma 4.4. *For matrices $G, H \in \mathbb{C}^{mn \times mr}$,*

$$\sum_{i=0}^{n-1} (X_0^T)^i G H^T X_0^i = \sum_{j=1}^r \mathbf{U}_j \mathbf{L}_j \quad \text{and} \quad (74)$$

$$\sum_{i=0}^{n-1} X_0^i G H^T (X_0^T)^i = \sum_{j=1}^r \check{\mathbf{L}}_j \check{\mathbf{U}}_j, \quad (75)$$

where \mathbf{U}_j is a block upper-triangular Toeplitz matrix with last block column $G\mathcal{E}_j$, \mathbf{L}_j is a block lower-triangular Toeplitz matrix with last block row $(H\mathcal{E}_j)^T$, $\check{\mathbf{L}}_j$ is a block lower-triangular Toeplitz matrix with last block row $(G\mathcal{E}_j)^P$, and $\check{\mathbf{U}}_j$ is a block upper-triangular Toeplitz matrix with last block column $\mathcal{J}_{m,n} H\mathcal{E}_j$.

Proof. See Appendix A. □

The Gohberg-Heinig formulas may then be connected to matrix displacement through the following theorem.

Theorem 4.2. *Let $T \in \mathbb{C}^{mn \times mn}$ be a nonsingular (m, n) two-level Toeplitz matrix and $B = T^{-1}$ its inverse. For any block column vector $Q \in \mathbb{C}^{mn \times m}$, let $\mathbf{L}(Q)$ be a block lower-triangular Toeplitz matrix with last block row Q^T and $\mathbf{U}(Q)$ be a block upper-triangular Toeplitz matrix with last block column Q . Let $G, H \in \mathbb{C}^{mn \times mr}$; then $B = \sum_{i=1}^r \mathbf{U}(G\mathcal{E}_i) \cdot \mathbf{L}(H\mathcal{E}_i)$ if and only if $\Delta_{X_0^T, X_0}(B) = GH^T$.*

Proof. Suppose $\Delta_{X_0^T, X_0}(B) = GH^T$; then from Lemma 1.1 and Lemma 4.4,

$$B = \sum_{i=1}^r \mathbf{U}(G\mathcal{E}_i) \cdot \mathbf{L}(H\mathcal{E}_i).$$

Conversely, if $B = \sum_{i=1}^r \mathbf{U}(G\mathcal{E}_i) \cdot \mathbf{L}(H\mathcal{E}_i)$, it follows from Lemma 4.4 that

$$B = \sum_{i=0}^{n-1} (X_0^T)^i GH^T X_0^i,$$

and thus

$$\begin{aligned} \Delta_{X_0^T, X_0}(B) &= \sum_{i=0}^{n-1} (X_0^T)^i GH^T X_0^i - X_0^T \left(\sum_{i=0}^{n-1} (X_0^T)^i GH^T X_0^i \right) X_0 \\ &= GH^T + (X_0^T)^n GH^T X_0^n = GH^T. \end{aligned}$$

□

Using Theorem 4.2, displacement can be used to generate Gohberg-Heinig formulas and vice versa. This result can be used to state *all* Gohberg-Heinig formulas.

Corollary 4.1. *Let $T \in \mathbb{C}^{mn \times mn}$ be a nonsingular (m, n) two-level Toeplitz matrix and $B = T^{-1}$ its inverse. Let $\mathbf{L}(\cdot)$ and $\mathbf{U}(\cdot)$ be defined as in Theorem 4.2, and $G, H \in \mathbb{C}^{mn \times mr}$. For a block diagonal matrix D with components $D_{i,i} \in \mathbb{C}^{m \times m}$, let the Gohberg-Heinig formula built from G , H , and D be denoted*

$$\mathfrak{G}_B(GDH^T) = \sum_{i=1}^r \mathbf{U}(G\mathcal{E}_i) \cdot (I_n \otimes D_{i,i}) \cdot \mathbf{L}(H\mathcal{E}_i). \quad (76)$$

Then $B = \mathfrak{G}_B(GDH^T)$ if and only if

$$GDH^T = \begin{bmatrix} \mathcal{U} & \mathcal{V}_{[0]} \end{bmatrix} \begin{bmatrix} I_m & 0 \\ 0 & -I_m \end{bmatrix} \begin{bmatrix} \mathcal{V}_{[0]}^P \\ \mathcal{U}^P \end{bmatrix}.$$

Proof. From Theorem 4.2, $B = \sum_{i=1}^r \mathbf{U}(G\mathcal{E}_i) \cdot (I_n \otimes D_{i,i}) \cdot \mathbf{L}(H\mathcal{E}_i)$ if and only if $\Delta_{X_0^T, X_0}(B) = GDH^T$. However, from Proposition 4.7,

$$\Delta_{X_0^T, X_0}(B) = \begin{bmatrix} \mathcal{U} & \mathcal{V}_{[0]} \end{bmatrix} \begin{bmatrix} I_m & 0 \\ 0 & -I_m \end{bmatrix} \begin{bmatrix} \mathcal{V}_{[0]}^P \\ \mathcal{U}^P \end{bmatrix}.$$

□

A more constructive statement of Corollary 4.1 is given as follows, though it restricts the Gohberg-Heinig forms to have only two summands.

Corollary 4.2. *Let $T \in \mathbb{C}^{mn \times mn}$ be a nonsingular (m, n) two-level Toeplitz matrix and $B = T^{-1}$ its inverse. Let $\mathbf{L}(\cdot)$ and $\mathbf{U}(\cdot)$ be defined as in Theorem 4.2. For any nonsingular $A \in \mathbb{C}^{2m \times 2m}$, written as a 2×2 block matrix $A = [A_{i,j}]$ with blocks satisfying*

$$A_{2,1}A_{2,2}^P = A_{2,2}A_{2,1}^P \quad \text{and} \quad A_{1,1}A_{1,2}^P = A_{1,2}A_{1,1}^P,$$

define

$$G = \begin{bmatrix} \mathcal{U} & \mathcal{V}_{[0]} \end{bmatrix} A^{-1} \quad H = \left(A^{-P} \begin{bmatrix} \mathcal{V}_{[0]}^P \\ \mathcal{U}^P \end{bmatrix} \right)^T.$$

Then

$$B = \mathbf{U}(G\mathcal{E}_1) \cdot D \cdot \mathbf{L}(H\mathcal{E}_1) - \mathbf{U}(G\mathcal{E}_2) \cdot D^{\mathcal{P}} \cdot \mathbf{L}(H\mathcal{E}_2),$$

where $D = I_n \otimes (A_{1,1}A_{2,2}^P - A_{1,2}A_{2,1}^P)$.

Proof. See Appendix A □

4.2.4 Equivalence of block-level displacements

Theorem 4.2 connects Gohberg-Heinig SSDs with displacement recovery. It also allows any pair of the block generators \mathcal{U} , $\mathcal{V}_{[R]}$, \mathcal{W} , and $\mathcal{X}_{[R]}$ to be defined with respect to any other pair.

Theorem 4.3. *For a nonsingular (m, n) two-level Toeplitz matrix $T = [A_{i-j}]$, let $B = T^{-1}$. Define the block column vectors \mathcal{U} , $\mathcal{V} = \mathcal{V}_{[R]}$, \mathcal{W} , and $\mathcal{X} = \mathcal{X}_{[S]}$ as in Proposition 4.5. Define also the related block column vectors $\dot{\mathcal{U}} = X_0^T \mathcal{U}$, $\dot{\mathcal{V}} = X_0^T - \mathcal{E}_n$, $\dot{\mathcal{W}} = X_0 \mathcal{W}$, and $\dot{\mathcal{X}} = X_0 \mathcal{X} - \mathcal{E}_1$. If the matrices \mathcal{X}_n , \mathcal{U}_1 , \mathcal{V}_1 , and $(I_m - \mathcal{V}_1 \mathcal{X}_n)$ are nonsingular, and*

$\Lambda = (I_m - \mathcal{V}_1 \mathcal{X}_n)^{-1} \mathcal{U}_1$, then the following relations hold:

$$\mathcal{U} = \mathcal{V} \mathcal{U}_1^P \mathcal{V}_1^{-P} + \dot{\mathcal{W}} \mathcal{V}_1^{-P}, \quad (77)$$

$$= -\mathcal{V} \mathcal{X}_n \Lambda - \dot{\mathcal{X}} \Lambda, \quad (78)$$

$$= \dot{\mathcal{W}} \mathcal{X}_n^P - \dot{\mathcal{X}} \mathcal{U}_1, \quad (79)$$

$$\mathcal{V} = \mathcal{U} \mathcal{V}_1^P \mathcal{U}_1^{-P} - \dot{\mathcal{W}} \mathcal{U}_1^{-P}, \quad (80)$$

$$= -\mathcal{U} (\mathcal{X}_n \Lambda)^{-P} - \dot{\mathcal{X}} \mathcal{X}_n^{-1}, \quad (81)$$

$$= -\dot{\mathcal{W}} \Lambda^{-P} - \dot{\mathcal{X}} \mathcal{V}_1, \quad (82)$$

$$\mathcal{W} = \dot{\mathcal{U}} \mathcal{V}_1^P - \dot{\mathcal{V}} \mathcal{U}_1^P, \quad (83)$$

$$= \dot{\mathcal{U}} \mathcal{X}_n^{-P} + \mathcal{X} \mathcal{U}_1 \mathcal{X}_n^{-P}, \quad (84)$$

$$= -\dot{\mathcal{V}} \Lambda^P - \mathcal{X} \Lambda \mathcal{V}_1^P, \quad (85)$$

$$\mathcal{X} = -\dot{\mathcal{U}} \Lambda^{-1} - \dot{\mathcal{V}} \mathcal{X}_n, \quad (86)$$

$$= -\dot{\mathcal{U}} \mathcal{U}_1^{-1} + \mathcal{W} \mathcal{X}_n^P \mathcal{U}_1^{-1}, \quad \text{and} \quad (87)$$

$$= -\dot{\mathcal{V}} \mathcal{V}_1^{-1} - \mathcal{W} \mathcal{V}_1^{-P} \Lambda^{-1}. \quad (88)$$

Proof. See Appendix A □

Combining Theorem 4.3 and Corollary 4.1, it is then possible to obtain a Gohberg-Heinig formula for any pair of generators that arise from the “natural” block-level displacements of Toeplitz inverses.

Theorem 4.4. *For a nonsingular (m, n) two-level Toeplitz matrix $T = [A_{i-j}]$, let $B = T^{-1}$. Define the block column vectors \mathcal{U} , $\dot{\mathcal{U}}$, \mathcal{V} , $\dot{\mathcal{V}}$, \mathcal{W} , $\dot{\mathcal{W}}$, \mathcal{X} , and $\dot{\mathcal{X}}$ as in Theorem 4.3. Assuming the matrices \mathcal{X}_n , \mathcal{U}_1 , \mathcal{V}_1 , and $(I_m - \mathcal{V}_1 \mathcal{X}_n)$ are nonsingular, let Λ be defined as in Theorem 4.3.*

Let the operators $\mathbf{L}(\cdot)$ and $\mathbf{U}(\cdot)$ be as given in Theorem 4.2 and the operator $\mathfrak{G}_B(\cdot)$ as in (76). Then the following are true:

$$1. \text{ If } G = \begin{bmatrix} \dot{\mathcal{U}} & \dot{\mathcal{V}} \end{bmatrix}, H = \begin{bmatrix} \mathcal{V}^P \\ \mathcal{U}^P \end{bmatrix}^T, \text{ and } D = \begin{bmatrix} I_m & \mathbf{0} \\ \mathbf{0} & -I_m \end{bmatrix}, \text{ then } B = \mathfrak{G}_B(GDH^T);$$

$$2. \text{ If } G = \begin{bmatrix} \dot{\mathcal{U}} & \mathcal{W} \end{bmatrix}, H = \begin{bmatrix} \dot{\mathcal{W}}^P \\ \mathcal{U}^P \end{bmatrix}^T, \text{ and } D = \begin{bmatrix} -\mathcal{U}_1^{-1} & \mathbf{0} \\ \mathbf{0} & \mathcal{U}_1^{-P} \end{bmatrix}, \text{ then } B = \mathfrak{G}_B(GDH^T);$$

$$3. \text{ If } G = \begin{bmatrix} \dot{\mathcal{U}} & \mathcal{X} \end{bmatrix}, H = \begin{bmatrix} \dot{\mathcal{X}}^P \\ \mathcal{U}^P \end{bmatrix}^T, \text{ and } D = \begin{bmatrix} -\mathcal{X}_n^{-P} & \mathbf{0} \\ \mathbf{0} & \mathcal{X}_n^{-1} \end{bmatrix}, \text{ then } B = \mathfrak{G}_B(GDH^T);$$

$$4. \text{ If } G = \begin{bmatrix} \mathcal{W} & \dot{\mathcal{V}} \end{bmatrix}, H = \begin{bmatrix} \mathcal{V}^P \\ \dot{\mathcal{W}}^P \end{bmatrix}^T, \text{ and } D = \begin{bmatrix} \mathcal{V}_1^{-P} & \mathbf{0} \\ \mathbf{0} & -\mathcal{V}_1^{-1} \end{bmatrix}, \text{ then } B = \mathfrak{G}_B(GDH^T);$$

$$5. \text{ If } G = \begin{bmatrix} \mathcal{X} & \dot{\mathcal{V}} \end{bmatrix}, H = \begin{bmatrix} \mathcal{V}^P \\ \dot{\mathcal{X}}^P \end{bmatrix}^T, \text{ and } D = \begin{bmatrix} -\Lambda & \mathbf{0} \\ \mathbf{0} & \Lambda^P \end{bmatrix}, \text{ then } B = \mathfrak{G}_B(GDH^T); \text{ and}$$

$$6. \text{ If } G = \begin{bmatrix} \mathcal{W} & \mathcal{X} \end{bmatrix}, H = \begin{bmatrix} \dot{\mathcal{X}}^P \\ -\dot{\mathcal{W}}^P \end{bmatrix}, \text{ and } D = \begin{bmatrix} -I_m & \mathbf{0} \\ \mathbf{0} & I_m \end{bmatrix}, \text{ then } B = \mathfrak{G}_B(GDH^T).$$

Proof. See Appendix A. □

4.2.5 Extension to blockwise results

An (m, n) two-level Toeplitz matrix can be permuted to become an (n, m) two-level Toeplitz matrix with the level-swapping matrix. Suppose T is an (m, n) two-level Toeplitz matrix, with blocks A_{i-j} having coefficients $[A_{i-j}]_{k,\ell} = a_{k-\ell, i-j}$. Let P be the (m, n) level-swapping matrix; then $T' = P^T T P$ has blocks $A'_{k-\ell}$ with coefficients $[A'_{k-\ell}]_{i,j} = a_{k-\ell, i-j}$, and is therefore also two-level Toeplitz. As a result, analogs of all of the block-level results can be obtained for blockwise formulas by swapping the matrix levels, invoking the theorems, and swapping the levels back to their original configuration.

4.3 Miscellaneous results on two-level Toeplitz inverses

Section 4.1 gave SSDs for special classes of two-level Toeplitz matrices, providing a first glimpse at multi-level Toeplitz inverse structure. Section 4.2 then broadened the class of matrices considered to generic two-level Toeplitz matrices, extending scalar results to

the two-level case on only a single structural level at once. Unfortunately, as has been repeatedly stated in literature, the extension of scalar Toeplitz theory and algorithms to generic multi-level Toeplitz matrices is an extraordinarily difficult problem, and one that is still very much open.

While there are many difficulties in extending scalar results to two-level Toeplitz matrices, the root cause of each is that two-level Toeplitz matrices are higher-dimensional structures that are condensed into a 2-D representation. The developments of Section 4.1 circumvent this problem by noting that when one or more structural levels of the matrix are commutative, the matrix still behaves as if it were a 1-D structure. The results of Section 4.2 instead essentially ignore a level of structure to treat the matrix as if it were indeed a 1-D structure. This treatment results in non-optimal results, but forges connections between the two-level inverse and known formulas for scalar Toeplitz matrices.

No closed-form, compact, and computationally useful description yet exists for generic two-level Toeplitz inverses. However, this research has produced a number of observations on known structural properties, conjectured decompositions, and potential descriptive generators. While these observations have not resulted in a closed-form decomposition for the two-level inverse, they do give insight into its structural properties and provide some guidance for further study.

4.3.1 On two-level displacement and displacement rank

The foundation of many scalar structured matrix inversion algorithms is the displacement-rank approach described in Section 1.4.1. It is then natural to consider extending this technique to the multi-level case by using displacement operators to exploit multiple levels of Toeplitz structure simultaneously. However, unlike the developments of Section 4.2, where Kronecker products were used to extend scalar Toeplitz matrix displacement to a single level, two-level displacement introduces a number of complications.

4.3.1.1 Composition of one-level displacements

An obvious approach to constructing a two-level displacement operator is to form it as the composition of two separate one-level displacements. Defining a blockwise shift matrix as $Y_f = Z_f \otimes \mathbf{I}$, then for $f, g \in \mathbb{C}$ this type of two-level displacement would be expressed as

$$\nabla^{(2)}(T) = \nabla_{Y_0, Y_g}(\nabla_{X_0, X_f}(T)) = Y_0 X_0 T - Y_0 T X_f - X_0 T Y_g + T X_f Y_g.$$

From their definitions, the matrices Y_g and X_f commute:

$$Y_g X_f = (Z_g \otimes \mathbf{I})(\mathbf{I} \otimes Z_f) = (Z_f \otimes Z_g) = (\mathbf{I} \otimes Z_f)(Z_g \otimes \mathbf{I}) = X_f Y_g.$$

Therefore, the order of the composition is irrelevant, and

$$\nabla^{(2)} = \nabla_{Y_0, Y_g} \circ \nabla_{X_0, X_f} = \nabla_{X_0, X_f} \circ \nabla_{Y_0, Y_g}.$$

From the commutativity of these types of matrices, a number of related two-level displacements can be defined that mix the various Sylvester and Stein displacements typically used for scalar Toeplitz matrices:

$$\begin{aligned} (\nabla_{X_0, X_0} \circ \nabla_{Y_0, Y_0}) &= (\nabla_{Y_0, Y_0} \circ \nabla_{X_0, X_0}) & (\nabla_{X_0^T, X_0^T} \circ \nabla_{Y_0, Y_0}) &= (\nabla_{Y_0, Y_0} \circ \nabla_{X_0^T, X_0^T}) \\ (\nabla_{X_0, X_0} \circ \nabla_{Y_0^T, Y_0^T}) &= (\nabla_{Y_0^T, Y_0^T} \circ \nabla_{X_0, X_0}) & (\nabla_{X_0^T, X_0^T} \circ \nabla_{Y_0^T, Y_0^T}) &= (\nabla_{Y_0^T, Y_0^T} \circ \nabla_{X_0^T, X_0^T}) \\ (\Delta_{X_0, X_0^T} \circ \nabla_{Y_0, Y_0}) &= (\nabla_{Y_0, Y_0} \circ \Delta_{X_0, X_0^T}) & (\Delta_{X_0^T, X_0} \circ \nabla_{Y_0, Y_0}) &= (\nabla_{Y_0, Y_0} \circ \Delta_{X_0^T, X_0}) \\ (\Delta_{X_0, X_0^T} \circ \nabla_{Y_0^T, Y_0^T}) &= (\nabla_{Y_0^T, Y_0^T} \circ \Delta_{X_0, X_0^T}) & (\Delta_{X_0^T, X_0} \circ \nabla_{Y_0^T, Y_0^T}) &= (\nabla_{Y_0^T, Y_0^T} \circ \Delta_{X_0^T, X_0}) \\ (\nabla_{X_0, X_0} \circ \Delta_{Y_0, Y_0^T}) &= (\Delta_{Y_0, Y_0^T} \circ \nabla_{X_0, X_0}) & (\nabla_{X_0^T, X_0^T} \circ \Delta_{Y_0, Y_0^T}) &= (\Delta_{Y_0, Y_0^T} \circ \nabla_{X_0^T, X_0^T}) \\ (\nabla_{X_0, X_0} \circ \Delta_{Y_0^T, Y_0}) &= (\Delta_{Y_0^T, Y_0} \circ \nabla_{X_0, X_0}) & (\nabla_{X_0^T, X_0^T} \circ \Delta_{Y_0^T, Y_0}) &= (\Delta_{Y_0^T, Y_0} \circ \nabla_{X_0^T, X_0^T}) \\ (\Delta_{X_0, X_0^T} \circ \Delta_{Y_0, Y_0^T}) &= (\Delta_{Y_0, Y_0^T} \circ \Delta_{X_0, X_0^T}) & (\Delta_{X_0^T, X_0} \circ \Delta_{Y_0, Y_0^T}) &= (\Delta_{Y_0, Y_0^T} \circ \Delta_{X_0^T, X_0}) \\ (\Delta_{X_0, X_0^T} \circ \Delta_{Y_0^T, Y_0}) &= (\Delta_{Y_0^T, Y_0} \circ \Delta_{X_0, X_0^T}) & (\Delta_{X_0^T, X_0} \circ \Delta_{Y_0^T, Y_0}) &= (\Delta_{Y_0^T, Y_0} \circ \Delta_{X_0^T, X_0}). \end{aligned}$$

Each of these composite displacements behaves slightly differently and has different properties on the two matrix levels. For simplicity, it is useful to consider the displacement of

two-level Toeplitz matrices relative to only one of these operators, as the results for the remainder are analogously obtained.

An analog to the final two points of Fact 1.1 may be given for the displacement $\nabla^{(2)}$.

Proposition 4.11. *Let $A \in \mathbb{C}^{mn \times mn}$ be written as an (m, n) two-level matrix. The following statements true:*

1. $(\nabla^{(2)}(A))^P = \nabla^{(2)}(A^P)$.
2. $(\nabla^{(2)}(A))^P = (\nabla^{(2)}(A^P))^{\mathcal{P}}$ and $(\nabla^{(2)}(A))^{\mathcal{P}} = (\nabla^{(2)}(A^P))^P$.
3. If $A = A^P$, then $(\nabla^{(2)}(A))^P = \nabla^{(2)}(A)$ and $(\nabla^{(2)}(A))^{\mathcal{P}} = (\nabla^{(2)}(A))^{\mathcal{P}}$.
4. If A is blockwise persymmetric, then $\nabla^{(2)}(A) = -(\nabla^{(2)}(A))^{\mathcal{P}}$. If A is block-level persymmetric, $\nabla^{(2)}(A) = -(\nabla^{(2)}(A))^P$. If A is fully persymmetric,

$$\nabla^{(2)}(A) = (\nabla^{(2)}(A))^P = -(\nabla^{(2)}(A))^{\mathcal{P}} = -(\nabla^{(2)}(A))^{\mathcal{P}}.$$

Proof. See Appendix A □

Unfortunately, the first three points of Fact 1.1 either do not hold or have no obvious analogs for two-level displacement. As a result, it is considerably harder to manipulate the displacement $\nabla^{(2)}$ than for the scalar displacement ∇_{Z_0, Z_0} .

In Section 4.2.2, it was shown that the nullspace of the displacement operator ∇_{X_0, X_0} is the set of all block lower-triangular Toeplitz matrices. Similar developments can be used to show that the nullspace of the displacement operator ∇_{Y_0, Y_0} is the set of all matrices with lower-triangular Toeplitz blocks. Therefore, the composition of these two displacement operators has a nullspace formed by the union of these two nullspaces.

4.3.1.2 Two-level displacement

Making use of the results of Section 4.2, one can obtain the following closed-form expression for the two-level Sylvester displacement of a two-level Toeplitz matrix.

Proposition 4.12. Let $T = [A_{i-j}]$ be (m, n) two-level Toeplitz with Toeplitz blocks $A_{i-j} \in \mathbb{C}^{m \times m}$ having coefficients $[A_{i-j}]_{k,\ell} = a_{i-j,k-\ell}$. Define

$$h_k := \left[0 \quad a_{k,1-m} \quad \cdots \quad a_{k,-1} \right]^T$$

and let

$$\mathbf{h} = \left[\mathbf{0} \quad h_{1-n} \quad \cdots \quad h_{-1} \right]^T, \quad \mathbf{H} = \begin{bmatrix} h_{-1} & \cdots & h_{1-n} & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathcal{E}_1 = I_n \otimes e_1.$$

Denoting the two-level Sylvester displacement operator as $\nabla^{(2)} = \nabla_{X_0, X_0} \circ \nabla_{Y_0, Y_0}$, then

$$\nabla^{(2)}(T) = \mathbf{h}e_1^P + e_1\mathbf{h}^P - (\mathbf{H}\mathcal{E}_1^P + \mathcal{E}_1\mathbf{H}^P). \quad (89)$$

Proof. See Appendix A □

A generic (m, n) two-level Toeplitz matrix is entirely parametrized by $(2m-1)(2n-1) < 4mn$ parameters. Since its matrix side length is mn , one would expect a displacement with good compression properties to yield a rank-4 expression for Toeplitz matrices. However, under the usual definition of rank, this is not quite the case.

The displacement does, however, compress the two-level Toeplitz structure down in a fashion analogous to scalar displacement, and the resulting displacement expression can be considered in terms of its two components. The first is the rank-2 sum $\mathbf{h}e_1^P + e_1\mathbf{h}^P$, and is similar in flavor to the displacement of a scalar Toeplitz matrix. The second is the term $\mathbf{H}\mathcal{E}_1^P + \mathcal{E}_1\mathbf{H}^P$, which satisfies

$$\text{rank}(\mathbf{H}\mathcal{E}_1^P + \mathcal{E}_1\mathbf{H}^P) \leq 2 \min(m-1, n-1).$$

Thus, the rank of this second term is considerably larger than would be desired.

The principal obstacle is that the concept of matrix rank does not take into account any multi-level structure. However, if the notion of rank is slightly relaxed, the term may still

be viewed as rank-2 in some generalized sense. Specifically,

$$\left(\mathbf{H}\mathcal{E}_1^P + \mathcal{E}_1\mathbf{H}^P\right)^P = \mathbf{h}e_1^P + e_1\mathbf{h}^P.$$

Therefore, the displacement can be thought of as the “generalized rank-4” expression

$$\nabla^{(2)}(T) = \mathbf{h}e_1^P + e_1\mathbf{h}^P - \left(\mathbf{h}e_1^P + e_1\mathbf{h}^P\right)^P.$$

Unfortunately, a precise mathematical description of this notion of generalized rank is difficult to obtain. While it is clear from the above expression that in some way the displacement $\nabla^{(2)}(T)$ is “rank-4,” it is unclear what definition of “rank” would reflect this structure and would be relevant for other two-level structured matrices.

4.3.1.3 Two-level Toeplitz inverse displacement

Aside from the problem of defining a generalized notion of rank, a significant roadblock to using two-level displacement to generate inverse formulas is connecting the two-level displacements of a matrix and its inverse. In the scalar case, this connection is straightforward because of the properties of Sylvester displacement. However, the nature of the composition of operators makes the same task difficult for two-level displacements.

For a nonsingular scalar matrix A , if the displacement of A satisfies $\nabla_{Z_0, Z_0}(A) = GH^T$ then $\nabla_{Z_0, Z_0}(A^{-1}) = -A^{-1}GH^T A^{-1}$. Therefore, it is evident that the displacement rank of A and A^{-1} are identical. Moreover, it is clear how to determine the generators of A^{-1} from those of A ; they are simply $A^{-1}G$ and $H^T A^{-1}$.

In the two-level case, however, the same does not hold:

$$\begin{aligned} A^{-1}\nabla^{(2)}(A)A^{-1} &= A^{-1}X_0Y_0 - A^{-1}X_0AY_0A^{-1} - A^{-1}Y_0AX_0^{-1} + X_0Y_0A^{-1} \\ &= \nabla^{(2)}(A^{-1}) + X_0A^{-1}Y_0 + Y_0A^{-1}X_0 - A^{-1}X_0AY_0A^{-1} - A^{-1}Y_0AX_0^{-1}. \end{aligned}$$

It is possible, however, to use the Sylvester displacement identity to define a relationship between the two-level displacements of a matrix and its inverse:

$$\begin{aligned} \nabla^{(2)}(A^{-1}) &= \nabla_{Y_0, Y_0}(\nabla_{X_0, X_0}(A^{-1})) = -\nabla_{Y_0, Y_0}(A^{-1}\nabla_{X_0, X_0}(A)A^{-1}) \\ &= Y_0A^{-1}\nabla_{X_0, X_0}(A)A^{-1} - A^{-1}\nabla_{X_0, X_0}(A)A^{-1}Y_0. \end{aligned}$$

Letting $Q = A^{-1}\nabla_{X_0,X_0}(A)A^{-1}$, it is clear that, using the standard definition of rank,

$$\text{rank}(Q) = \text{rank}(\nabla_{X_0,X_0}(A)).$$

Therefore, the best that can be said about the rank of the displacement $\nabla^{(2)}(A^{-1})$ is

$$\text{rank}(\nabla^{(2)}(A^{-1})) \leq 2 \cdot \text{rank}(\nabla^{(2)}(A)).$$

Similar manipulation of the second point of Fact 1.1 can be used to show the following:

$$\begin{aligned} -\nabla^{(2)}(A^{-1}) &= A^{-1}\nabla^{(2)}(A)A^{-1} + A^{-1}\nabla_{X_0,X_0}(A)\nabla_{Y_0,Y_0}(A^{-1}) + A^{-1}\nabla_{Y_0,Y_0}(A)\nabla_{X_0,X_0}(A^{-1}) \\ &= A^{-1}\nabla^{(2)}(A)A^{-1} + \nabla_{X_0,X_0}(A^{-1})\nabla_{Y_0,Y_0}(A)A^{-1} + \nabla_{Y_0,Y_0}(A^{-1})\nabla_{X_0,X_0}(A)A^{-1}. \end{aligned}$$

These equations indicate that the relationship between the two-level displacements of a matrix and its inverse are significantly more complicated than the scalar displacement relationship.

4.3.2 On two-level Gohberg-Semencul formulas

The Gohberg-Semencul formulas decompose Toeplitz matrices and inverses as sums of products of triangular Toeplitz matrices. It is then reasonable to wonder when considering two-level Toeplitz matrices whether similar decompositions exist. Unfortunately, this has proven to be a remarkably difficult question to answer.

For scalar matrices, the displacement-rank approach can be used to prove the existence of these decompositions, as is shown in Lemma 2 of [63]. However, as discussed in Section 4.3.1.3, there are no obvious two-level displacement operators that yield low displacement rank for the two-level inverse. As a result, the displacement-rank approach is presently unavailable for demonstrating that two-level Gohberg-Semencul formulas exist.

Alternatively, a polynomial interpretation of Toeplitz matrices and inverses relative to so-called “generating functions” can be used to prove the existence of Gohberg-Semencul formulas for scalar matrices. This general approach was extended to multivariate polynomials and generators for multi-level functions in [79]. The result of this work was a

definition of the generating function of a “multivariate Bézoutian” (multi-level Toeplitz inverse), but this polynomial has no obvious connection to a two-level Gohberg-Semencul decomposition.

Still, there are several contributions that can be made regarding postulates of two-level Gohberg-Semencul formulas. Given the structure of the scalar Gohberg-Semencul formulas, there are several forms that one might immediately suspect that a two-level Gohberg-Semencul might take. Using what properties are known of the two-level inverse, it can be shown that many of these forms are equivalent.

4.3.2.1 Persymmetry properties of two-level inverses

Two-level Toeplitz matrices are persymmetric, but can also be considered to be *two-level persymmetric*, meaning they exhibit persymmetry both on the block level and in their individual blocks. This property implies the following relations regarding the inverse of a nonsingular two-level Toeplitz matrix.

Proposition 4.13. *Let T be a nonsingular (m, n) two-level Toeplitz matrix, $B = T^{-1}$ its inverse, and denote $\mathcal{J}_{m,n} = I_n \otimes J_m$ and $\mathcal{J}_{m,n} = J_n \otimes I_m$. Then the following relations hold:*

$$T^{-1} = B^P \quad (90)$$

$$(T^{\mathcal{T}})^{-1} = \mathcal{J}_{m,n} B \mathcal{J}_{m,n} = \mathcal{J}_{m,n} (T^T)^{-1} \mathcal{J}_{m,n} \quad (91)$$

$$(T^{\mathcal{T}})^{-1} = \mathcal{J}_{m,n} B \mathcal{J}_{m,n} = \mathcal{J}_{m,n} (T^T)^{-1} \mathcal{J}_{m,n} \quad (92)$$

Proof. It is obvious that (90) follows from the definition of persymmetry. For (91),

$$T^{\mathcal{T}} = \mathcal{J}_{m,n} T \mathcal{J}_{m,n} \Rightarrow (T^{\mathcal{T}})^{-1} = \mathcal{J}_{m,n} B \mathcal{J}_{m,n}$$

and

$$J_{mn} T^{-T} J_{mn} = (J_{mn} T^T J_{mn})^{-1} = T^{-1} = B \Rightarrow \mathcal{J}_{m,n} B \mathcal{J}_{m,n} = \mathcal{J}_{m,n} (T^T)^{-1} \mathcal{J}_{m,n}.$$

Similar manipulations are used to show (92). □

Proposition 4.13 indicates that, save for special cases, the two-level inverse is not itself block and blockwise persymmetric. Indeed, the inverse B is only two-level persymmetric if $(T^{-1})^{\mathcal{T}} = (T^{\mathcal{T}})^{-1}$ and $(T^{-1})^{\mathcal{T}} = (T^{\mathcal{T}})^{-1}$, which is not generally the case.³

4.3.2.2 Four-term formula postulates

The component matrices of the scalar Gohberg-Semencul formulas are triangular Toeplitz matrices. Since the matrices are scalar, there are only two options for the triangularity of these components (lower or upper), and the ordering of the multiplication is irrelevant so long as all terms follow the same convention. For example, a persymmetric matrix whose generators are the sets $\{x_i, y_i\}$ satisfies

$$A = \sum_{i=1}^r L(x_i) \cdot U(y_i) = \sum_{i=1}^r U(y_i) \cdot L(x_i).$$

The invariance of the ordering, as shown as in Proposition 2.8, is due to the persymmetry of A .

A reasonable suggestion for components of a potential two-level Gohberg-Semencul formula are two-level-triangular two-level Toeplitz matrices (*i.e.*, two-level Toeplitz matrices that are triangular in both levels). However, the ordering of terms for this type of formula is *not* invariant, as the two-level inverse is not two-level persymmetric. Regardless, using the persymmetry properties of Section 4.3.4 it can be shown that many orderings are equivalent.

Let $LU(x)$ denote an (m, n) two-level Toeplitz matrix that has a lower-triangular block pattern and upper-triangular blocks and is parametrized by the vector $x \in \mathbb{C}^{mn \times 1}$, and $LL(x)$, $UL(x)$, and $UU(x)$ be defined similarly. One may consider as potential Gohberg-Semencul formulas sums of the following 24 forms:

$$(F_1) : \sum_i LL(u_i)LU(v_i)UL(w_i)UU(x_i) \quad (F_2) : \sum_i LL(u_i)LU(v_i)UU(w_i)UL(x_i)$$

³This is an example of a property that one-level transposes do not share with the general transpose operator.

$$\begin{aligned}
(F_3) : & \sum_i LL(u_i)UL(v_i)LU(w_i)UU(x_i) & (F_4) : & \sum_i LL(u_i)UL(v_i)UU(w_i)LU(x_i) \\
(F_5) : & \sum_i LL(u_i)UU(v_i)LU(w_i)UL(x_i) & (F_6) : & \sum_i LL(u_i)UU(v_i)UL(w_i)LU(x_i) \\
(F_7) : & \sum_i LU(u_i)LL(v_i)UL(w_i)UU(x_i) & (F_8) : & \sum_i LU(u_i)LL(v_i)UU(w_i)UL(x_i) \\
(F_9) : & \sum_i LU(u_i)UL(v_i)LL(w_i)UU(x_i) & (F_{10}) : & \sum_i LU(u_i)UL(v_i)UU(w_i)LL(x_i) \\
(F_{11}) : & \sum_i LU(u_i)UU(v_i)LL(w_i)UL(x_i) & (F_{12}) : & \sum_i LU(u_i)UU(v_i)UL(w_i)LL(x_i) \\
(F_{13}) : & \sum_i UL(u_i)LL(v_i)LU(w_i)UU(x_i) & (F_{14}) : & \sum_i UL(u_i)LL(v_i)UU(w_i)LU(x_i) \\
(F_{15}) : & \sum_i UL(u_i)LU(v_i)LL(w_i)UU(x_i) & (F_{16}) : & \sum_i UL(u_i)LU(v_i)UU(w_i)LL(x_i) \\
(F_{17}) : & \sum_i UL(u_i)UU(v_i)LL(w_i)LU(x_i) & (F_{18}) : & \sum_i UL(u_i)UU(v_i)LU(w_i)LL(x_i) \\
(F_{19}) : & \sum_i UU(u_i)LL(v_i)LU(w_i)UL(x_i) & (F_{20}) : & \sum_i UU(u_i)LL(v_i)UL(w_i)LU(x_i) \\
(F_{21}) : & \sum_i UU(u_i)LU(v_i)LL(w_i)UL(x_i) & (F_{22}) : & \sum_i UU(u_i)LU(v_i)UL(w_i)LL(x_i) \\
(F_{23}) : & \sum_i UU(u_i)UL(v_i)LL(w_i)LU(x_i) & (F_{24}) : & \sum_i UU(u_i)UL(v_i)LU(w_i)LL(x_i).
\end{aligned}$$

These forms can be categorized into three basic groups by the number of similar level structures that adjacent multiplicative terms share. For example, for the terms in form F_1 , the block-level triangularity of the multiplicative terms $LL(u_i)LU(v_i)$ is the same, but the triangularity of the blocks is different. The terms of the product $LU(v_i)UL(w_i)$ share no similarity in the triangularity of their levels, while the term $UL(w_i)UU(x_i)$ again shares block-level structure. Therefore, F_1 is referred to as being in the $(1, 0, 1)$ group. Then there are three groups:

$$\begin{aligned}
(1, 0, 1) : & F_1, F_3, F_8, F_{11}, F_{14}, F_{17}, F_{22}, F_{24} \\
(1, 1, 1) : & F_2, F_4, F_7, F_{12}, F_{13}, F_{18}, F_{21}, F_{23} \\
(0, 1, 0) : & F_5, F_6, F_9, F_{10}, F_{15}, F_{16}, F_{19}, F_{20}.
\end{aligned}$$

Using the persymmetry properties of Section 4.3.2.1, one can show that if any form of one of these groups is a valid Gohberg-Semencul formula, the remainder of the forms in the

group must be as well.

Beginning with the $(1, 0, 1)$ group, since two-level Toeplitz matrices are persymmetric, their inverses are as well. Taking the persymmetric transpose of any individual form results in a reversal of the structures. For example, if

$$B = \sum_i LL(u_i)LU(v_i)UL(w_i)UU(x_i)$$

then it follows that

$$B = B^P = \left(\sum_i LL(u_i)LU(v_i)UL(w_i)UU(x_i) \right)^P = \sum_i UU(x_i)UL(w_i)LU(v_i)LL(u_i),$$

and therefore forms F_1 and F_{24} are equivalent. Thus, from the $(1, 0, 1)$ forms, the pairs (F_1, F_{24}) , (F_3, F_{22}) , (F_8, F_{17}) and (F_{11}, F_{14}) are equivalent structures.

Next, let T be (m, n) two-level Toeplitz; since $T^{\mathcal{T}}$ is also two-level Toeplitz its inverse must have a Gohberg-Semencul formula of the same form. From Propostion 4.13, then,

$$\begin{aligned} (T^{\mathcal{T}})^{-1} &= \mathcal{I}_{m,n} \left(\sum_i LL(u_i)LU(v_i)UL(w_i)UU(x_i) \right) \mathcal{I}_{m,n} \\ &= \sum_i \mathcal{I}_{m,n} LL(u_i) \mathcal{I}_{m,n}^2 LU(v_i) \mathcal{I}_{m,n}^2 UL(w_i) \mathcal{I}_{m,n}^2 UU(x_i) \mathcal{I}_{m,n} \\ &= \sum_i LU(\tilde{u}_i)LL(\tilde{v}_i)UU(\tilde{w}_i)UL(\tilde{x}_i). \end{aligned}$$

Since this would be a valid SSD, it implies that any form F_1 must have an equivalent form F_8 , and therefore the two sets of forms $(F_1, F_{24}, F_8, F_{17})$ and $(F_3, F_{22}, F_{11}, F_{14})$ contain equivalent structures.

Finally, the (m, n) level-swapping matrix can be used to exchange level structures. Since the matrix P^TTP is also a two-level Toeplitz matrix, it follows that if form F_1 is a valid Gohberg-Semencul formula then

$$\begin{aligned} P^TTP &= P^T \left(\sum_i LL(u_i)LU(v_i)UL(w_i)UU(x_i) \right) P \\ &= \sum_i P^T LL(u_i)P \cdot P^T LU(v_i)P \cdot P^T UL(w_i)P \cdot P^T UU(x_i)P \\ &= \sum_i LL(\tilde{u}_i)UL(\tilde{v}_i)LU(\tilde{w}_i)UU(\tilde{x}_i), \end{aligned}$$

and so there must also be an equivalent form F_3 . Therefore, the set of forms $(F_1, F_3, F_8, F_{11}, F_{14}, F_{17}, F_{22}, F_{24})$ are all equivalent.

Similar manipulations show that the same follows for the $(1, 1, 1)$ forms and the $(0, 1, 0)$ forms. While it has not been shown that a four-term expression of the forms F_1 – F_{24} exists for the two-level inverse, from persymmetry and two-level structure arguments it can at least be shown that many of these forms are equivalent. Therefore, as in the scalar case, if such a form is discovered it will immediately imply that many equivalent forms exist.

4.3.2.3 Two-term formula postulates

Another reasonable proposal for a potential two-level Gohberg-Semencul formula involves only two multiplicative terms in each summand. Considering that more parameters define two-level Toeplitz matrices than scalar Toeplitz matrices, one would expect such formulas to involve more summands than in the scalar case. Using similar developments as in Section 4.3.2.2, it is possible to state restrictions on potential formulas on this type.

First, if the component matrices are two-level-triangular two-level Toeplitz matrices, then the formulas do not consist of only a few summands. For instance, a product of the form $LL(u)UU(v)$, where $LL(u)$ has a first column of u and $UU(v)$ has a last column of $J_{m,n}v$. This product may equivalently be written as

$$LL(u)UU(v) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} X_0^i Y_0^j u v^T (X_0^T)^{i-1} (Y_0^T)^{j-1}.$$

Since both X_0 and Y_0 are nilpotent, from Theorem 1.1 if a two-level Toeplitz inverse B could be written as

$$B = \sum_{i=1}^r LL(u_i)UU(v_i),$$

then $\Delta_{X_0 Y_0, X_0^T Y_0^T}(B) = UV^T$, where u_i are the columns of U and v_i are the columns of V .

Similar expressions can be shown for other two-level triangular products. If any of these formulas contained only a few summands, it would necessarily imply that two-level Toeplitz inverses have low two-level Stein displacement rank. As discussed in Section 4.3.1, this is not in general the case.

Therefore, if there exist formulas with only a small number of summands with two multiplicative terms, the structures of the terms must exhibit some variation. Since there are four possible two-level triangular structures, there are twelve possible structured products:

$$\begin{array}{ll}
(F_{25}) : LL(u_i)LU(v_i) & (F_{26}) : LL(u_i)UL(v_i) \\
(F_{27}) : LL(u_i)UU(v_i) & (F_{28}) : LU(u_i)LL(v_i) \\
(F_{29}) : LU(u_i)UL(v_i) & (F_{30}) : LU(u_i)UU(v_i) \\
(F_{31}) : UL(u_i)LL(v_i) & (F_{32}) : UL(u_i)LU(v_i) \\
(F_{33}) : UL(u_i)UU(v_i) & (F_{34}) : UU(u_i)LL(v_i) \\
(F_{35}) : UU(u_i)LU(v_i) & (F_{36}) : UU(u_i)UL(v_i).
\end{array}$$

Several observations can be made regarding the number of terms of the various forms using the same persymmetry arguments as in Section 4.3.2.2.

Suppose a valid two-level Gohberg-Semencul formula contains r_i summands of form F_i . Then by the persymmetry of the two-level inverse, there exists an equivalent Gohberg-Semencul formula containing r_{28} summands of form F_{25} , r_{31} summands of form F_{26} , and so on. Similarly, since the matrices T^T , $T^{\mathcal{S}}$, P^TTP are also two-level Toeplitz, it follows that there must also exist equivalent Gohberg-Semencul formulas with r_{33} , r_{28} , and r_{26} summands of form F_{25} and so on. Table 9 summarizes these equivalences.

Unlike the four-term case, since two-level Stein displacement can be used to generate formulas containing only a single form, it is provable that such formulas exist for the two-level inverse. In these cases, the formulas consist of a single r_i that is non-zero (and thus the formula can be written in an equivalent form with a different structured product). However, as stated before, these formulas do not consist of a small number of terms, but rather have $\mathcal{O}(\max(m, n))$ terms as stated (for Sylvester displacement) in Section 4.3.1. Since these formulas result in $\mathcal{O}(\max(m, n)mn)$ total parameters defining the inverse, they are impractical.

Table 9: Number of each form of structured product for equivalent potential Gohberg-Semencul-type formulas for two-level Toeplitz inverses. Assuming the existence of a Gohberg-Semencul formula containing the specified number of terms of each form as listed in the second column, there must also exist equivalent formulas with the specified number of terms in each of the remaining columns.

Form	Original (assumed)	Equivalence 1 (B^p)	Equivalence 2 ($\mathcal{J}B\mathcal{J}$)	Equivalence 3 ($\mathcal{J}B\mathcal{J}$)	Equivalence 4 (P^TBP)
F_{25}	r_{25}	r_{28}	r_{33}	r_{28}	r_{26}
F_{26}	r_{26}	r_{31}	r_{31}	r_{30}	r_{25}
F_{27}	r_{27}	r_{34}	r_{32}	r_{29}	r_{27}
F_{28}	r_{28}	r_{25}	r_{36}	r_{25}	r_{31}
F_{29}	r_{29}	r_{32}	r_{34}	r_{27}	r_{32}
F_{30}	r_{30}	r_{35}	r_{35}	r_{26}	r_{33}
F_{31}	r_{31}	r_{26}	r_{26}	r_{35}	r_{28}
F_{32}	r_{32}	r_{29}	r_{27}	r_{34}	r_{29}
F_{33}	r_{33}	r_{36}	r_{25}	r_{36}	r_{30}
F_{34}	r_{34}	r_{27}	r_{29}	r_{32}	r_{34}
F_{35}	r_{35}	r_{30}	r_{30}	r_{31}	r_{36}
F_{36}	r_{36}	r_{33}	r_{28}	r_{33}	r_{35}

Table 9 does convey, however, that if any formula were found containing a small number of terms of mixed form, several other related decompositions would follow immediately. Such information also makes the search for such forms simpler, as the equivalences are evident. To date, however, no such results appear to exist.

4.3.3 On two-level Toeplitz inverse generators

In Section 2.1.1.1, several sets of generators for scalar Toeplitz inverses were derived through displacement equations. Subsequently, in Section 2.1.1.4, the equivalence of any pair of these generators was shown. Together, these results convey the fact that the Toeplitz inverse is completely described by a pair of linearly independent vectors from a vector space of dimension 2. Such a result is unsurprising, since it should be expected that a total of $2n$ parameters uniquely determine a scalar Toeplitz inverse.

Another natural question of two-level structure is whether the two-level inverse can be similarly described with a linear subspace of dimension 4. A positive answer would seem intuitive, as a two-level Toeplitz inverse *should* be entirely be described with only $(2m -$

$1)(2n - 1)$ parameters. However, since displacement does not yield low rank expressions and since no known compact two-level Gohberg-Semencul forms are known, the question has yet to be definitively answered. In the words of Heinig [54],

...the question is where these parameters are hidden and how the whole inverse matrix can be recovered from them.

4.3.3.1 *Columns as generators*

It is worthwhile, however, to remark on a few possibilities and state some facts regarding potential generators for the two-level Toeplitz inverse. In the scalar case, one pair of generators that is particularly convenient and straightforward consists of the first and last columns of the Toeplitz inverse. Indeed, this pair of generators was the foundation for the original Gohberg-Semencul formula.

Inverse formulas that use specific columns of the inverse are appealing, as their generalizations to the two-level case would seem more intuitive. For instance, it is a reasonable conjecture that if there are scalar inverse formulas built upon specific columns of the inverse, the same might be true for two-level Toeplitz inverses. By contrast, it is less clear what the proper extensions of the vectors v and x in Proposition 2.4 would be for the two-level case.

Inverse descriptions that are built entirely from a subset of columns of the inverse are then of particular interest. Indeed, a considerable amount of effort has been spent in determining precisely *which* sets of columns of a Toeplitz inverse suffice. Gohberg and Krupnik gave conditions under which the first two columns were sufficient [45], while Ben-Artzi and Shalom later proved that any combination of three columns was enough to build the inverse [10].

Labahn and Shalom later improved this result [68]. Their work states that once the first column $u = Be_1$ of the inverse is known, at most only a single additional column of the inverse is needed. Moreover, which specific columns may be used can be determined

directly from the entries of u . However, their reconstruction formulas also require the coefficients of the Toeplitz matrix; while these coefficients are known, it in some sense results in the inverse being parametrized by $(3n - 2)$, rather than $(2n - 1)$, parameters. This deficiency was later removed in a related modified result by Heinig [54].

What is interesting about formulas relying on exactly two columns of the inverse is that they perfectly capture *the exact* number of degrees of freedom in the inverse. Since the inverse is persymmetric, any pair of columns will have at least one entry in common, and therefore there are at most $(2n - 1)$ true degrees of freedom in the pair. Since this is also the number of coefficients that define a scalar Toeplitz matrix, it seems to be an intuitive result.

Unfortunately, for general two-level Toeplitz matrices the inverse is not two-level persymmetric. Therefore, any set of four columns will have $(4mn - 4)$, rather than $(2m - 1)(2n - 1) = (4mn - 2(m + n) + 1)$ unique entries. This characteristic does not rule out the possibility that a set of four columns might yield a two-level Gohberg-Semencul formula, but it does suggest that such a formula would not have as compact and obvious a representation of the degrees of freedom in the problem as the scalar versions.

4.3.3.2 More general submatrices as generators

While a subset of columns may not produce intuitive generators for the two-level inverse, one may consider instead broadening the scope to analogous submatrices of the two-level inverse that better reflect the multi-level structure at hand. To better explain, consider the original Gohberg-Semencul formula, which depends on the first and last columns of the inverse. This formula may alternatively be interpreted as depending on the first column and first row of the inverse, as $e_1^T B = (B e_n)^P$. Under this line of thought, one could consider four possible generators for the two-level inverse:

$$\begin{aligned} u &= (I_m \otimes I_n)B(e_1 \otimes e_1) & v &= (I_n \otimes e_1^T)B(e_1 \otimes I_m) \\ w &= (e_1^T \otimes I_m)B(I_n \otimes e_1) & x &= (e_1^T \otimes e_1^T)B(I_n \otimes I_m). \end{aligned} \tag{93}$$

Table 10: Number of common entries between pairs of the potential two-level inverse generators defined in (93). For each pair of generators, the entries $B_{i,j}$ they share in common are listed.

	u	v	w	x
u	–	$B_{1+(j-1)m,1}$	$B_{i,1}$	$B_{1,1}$
v	$B_{1+(j-1)m,1}$	–	$B_{1,1}$	$B_{1,i}$
w	$B_{i,1}$	$B_{1,1}$	–	$B_{1,1+(j-1)m}$
x	$B_{1,1}$	$B_{1,i}$	$B_{1,1+(j-1)m}$	–

The components u and x correspond to the first column and first row/last column, respectively. The component v is composed of the first rows of each block in the first block column (and is therefore $n \times m$) while the component w is composed of the first columns of each of the blocks in the first block row (and is therefore $m \times n$). The generators v and w are not columns of the inverse, but they are still submatrices of the inverse containing a total of mn coefficients.

What is appealing about this choice of potential generators is that an inspection of the shared entries of its components reveals that the number of free parameters corresponds to the number of free parameters in a two-level Toeplitz matrix. The common entries shared between each pair of generators is shown in Table 10. The total number of unique entries can then be taken as the sum of the generator sizes minus the number of shared entries. Using Table 10 and ignoring repeated shared entries,⁴ the following is a count of the total number of free parameters:

$$\begin{aligned} \text{No. parameters} &= 4mn - n - m - m - n + 1 \\ &= 4mn - 2(m + n) + 1 = (2m - 1)(2n - 1). \end{aligned}$$

Therefore, the set $\{u, v, w, x\}$ contains exactly as many unique entries as free parameters in a two-level Toeplitz matrix.

⁴For example, w shares $B_{1,1}$ with both u and v , so this entry should not be subtracted twice from the number of unique entries in the set $\{u, v, w, x\}$.

4.3.3.3 Connections to displacement

Another aspect that makes the generators of Section 4.3.3.2 appealing as potential candidates are the connections they have to terms that appear in the two-level displacement of two-level Toeplitz matrices. For scalar matrices, (21) and (22) give the Sylvester displacement of the scalar Toeplitz inverse depending on the vectors u and w , which can be defined as $u = Be_1$ and $w^P = e_1^T B$. It is possible to show that the potential generators $\{u, v, w, x\}$ of (93) can be expressed in a similar way.

In Section 4.3.1, the two-level displacement of a two-level Toeplitz matrix was shown to be given by (89). Much as in Proposition 2.1, there are two terms of this expression that do not depend on the entries of T : the vector e_1 and the matrix $\mathcal{E}_1 = I_n \otimes e_1$. This expression could also equivalently be written in terms of the products $\widetilde{H}\mathcal{E}_1^P + \mathcal{E}_1\widetilde{H}^P$, where $\mathcal{E}_1 = e_1 \otimes I_m$ and \widetilde{H} is a $mn \times m$ matrix.

The connection between the potential generators $\{u, v, w, x\}$ and displacement expressions is now clear. From the definition of the generators,

$$\begin{aligned} u &= Be_1 & v &= \mathcal{E}_1^T B \mathcal{E}_1 \\ w &= \mathcal{E}_1^T B \mathcal{E}_1 & x &= e_1^T B. \end{aligned}$$

Therefore, each generator can be written in a similar form as the scalar generators of Proposition 2.4.

Since two-level displacement is difficult to work with, it has yet to be shown that the generators of (93) are indeed generators for the two-level inverse. However, their connection to scalar expressions and the number of free parameters among them make it a particularly intriguing conjecture that there exists a decomposition for the two-level inverse built from their entries. If such a form exists, it would serve as an analog to the Gohberg-Semencul for two-level matrices.

4.3.4 On transformation into two-level Loewner matrices

As described in Section 1.3, one set of scalar superfast Toeplitz inversion algorithms operates by transforming the matrix with Fourier-like operators to deal instead with a Loewner matrix. The same approach can be taken with two-level Toeplitz matrices, where the transformations involved are 2-D Fourier-like operators. As a result, one may work with two-level Loewner matrices instead of two-level Toeplitz matrices

4.3.4.1 Generalized block-Cauchy matrices

To establish how two-level Toeplitz matrices can be transformed into different structures, it is necessary to first extend the definition of generalized Cauchy matrices to multi-level versions, beginning with block forms.

Definition 4.3 (Generalized block Cauchy matrices). *For fixed $r, M, N \in \mathbb{N}$ and $i, j = 1, \dots, N$, let $U_i, V_j \in \mathbb{C}^{rM \times M}$ and $C_i, D_j \in \mathbb{C}^{M \times M}$. Further, let C_i and D_j satisfy the following for all $i, j = 1, \dots, N$:*

- $(C_i - D_j)^{-1}$ exists.
- $C_i(U_i^T V_j) = (U_i^T V_j)C_i$.
- $C_i(C_i - D_j)^{-1} = (C_i - D_j)^{-1}C_i$.
- $D_j(U_i^T V_j) = (U_i^T V_j)D_j$.
- $D_j(C_i - D_j)^{-1} = (C_i - D_j)^{-1}D_j$.
- $(U_i^T V_j)(C_i - D_j)^{-1} = (C_i - D_j)^{-1}(U_i^T V_j)$.

Then the matrix $\Phi = [\Phi_{i,j}]_{i,j=1}^N \in \mathbb{C}^{MN \times MN}$ with blocks

$$\Phi_{i,j} = U_i^T V_j (C_i - D_j)^{-1} \quad (94)$$

is a **generalized block Cauchy matrix**.

For a sequence of matrices $\Sigma = \{\Sigma_i\}$, with $\Sigma_i \in \mathbb{C}^{M \times M}$, define the block diagonalizing operator $\mathcal{D}(\cdot)$ as

$$\mathcal{D}(\Sigma) = \begin{bmatrix} \Sigma_1 & & & \\ & \Sigma_2 & & \\ & & \ddots & \\ & & & \Sigma_N \end{bmatrix} \in \mathbb{C}^{MN \times MN}. \quad (95)$$

Using this operation, the following proposition holds.

Proposition 4.14. *Let Φ be a generalized block-Cauchy matrix as in Definition 4.3. The block Sylvester displacement of Φ is*

$$\mathcal{D}(C) \Phi - \Phi \mathcal{D}(D) = U^T V, \quad (96)$$

where $C = \{C_i\}_{i=1}^N$, $D = \{D_j\}_{j=1}^N$, and U and V are such that $(U^T V)_{i,j} = U_i^T V_j$.

Proof.

$$\begin{aligned} [\mathcal{D}(C) \Phi - \Phi \mathcal{D}(D)]_{i,j} &= C_i U_i^T V_j (C_i - D_j)^{-1} - U_i^T V_j (C_i - D_j)^{-1} D_j \\ &= U_i^T V_j C_i (C_i - D_j)^{-1} - U_i V_j D_j (C_i - D_j)^{-1} \\ &= U_i^T V_j (C_i - D_j) (C_i - D_j)^{-1} = U_i^T V_j. \end{aligned}$$

□

A converse results holds as well.

Proposition 4.15. *Given $U_i, V_j \in \mathbb{C}^{rM \times M}$, $C_i, D_j \in \mathbb{C}^{M \times M}$ for $i, j = 1, \dots, N$ such that the necessary commutativity and inversion conditions hold, Φ is a generalized block Cauchy matrix if (96) holds and C_i and D_j commute with $\Phi_{i,j}$.*

Proof. Assume (96) is true; then the $(i, j)^{th}$ block of the matrix is

$$U_i^T V_j = [\mathcal{D}(C) \Phi - \Phi \mathcal{D}(D)]_{i,j} = C_i \Phi_{i,j} - \Phi_{i,j} D_j = \Phi_{i,j} (C_i - D_j).$$

As a consequence, the $(i, j)^{th}$ block of Φ can be written as

$$\Phi_{i,j} = Z_i^T Y_j (C_i - D_j)^{-1},$$

and so Φ is a generalized block-Cauchy matrix. \square

4.3.4.2 Displacement and block-level transformation

Proposition 4.4 gave an expression for the block-level displacement of a two-level Toeplitz matrix. One may also consider the following displacement, which has the advantage that each of its component matrices is diagonalizable and invertible.

Proposition 4.16. *Let $T = [A_{i-j}]$ be (m, n) two-level Toeplitz with blocks A_{i-j} . Then*

$$\nabla_{X_{-1}, X_1}(T) = X_{-1}T - TX_1 = GH^T, \quad (97)$$

where

$$G = \begin{bmatrix} A_0 & -I_m \\ (A_1 - A_{1-n}) & \mathbf{0} \\ \vdots & \vdots \\ (A_{n-1} - A_{-1}) & \mathbf{0} \end{bmatrix} \quad \text{and} \quad H = \begin{bmatrix} \mathbf{0} & (A_{n-1} + A_{-1})^T \\ \vdots & \vdots \\ \mathbf{0} & (A_1 + A_{1-n})^T \\ -I_m & A_0^T \end{bmatrix}. \quad (98)$$

Proof. The proof proceeds in an identical fashion to that of Proposition 4.4. \square

In the proof of Lemma 4.2, it was shown that for $f = \mathbf{e}^{j\theta}$, with $\theta \in (-\pi, \pi]$,

$$Z_f = \bar{\Lambda}_f \mathbf{F}^H (\mathbf{e}^{j\theta/n} \Sigma) \mathbf{F} \Lambda_f,$$

where $\Sigma = \text{diag}(\mathbf{e}^{-j2\pi k/n})$ and $\Lambda_f = \text{diag}(\mathbf{e}^{j\theta k/n})$. For $f = 1$, it follows that

$$Z_1 = \mathbf{F}^H \Sigma \mathbf{F}.$$

For $f = -1$, $\Lambda_{-1} = \text{diag}(\mathbf{e}^{j\pi k/n})$ and $\mathbf{e}^{j\pi/n} \Sigma = \text{diag}(\mathbf{e}^{-j(2k-1)\pi/n})$. These developments lead to the following block-level extensions.

Proposition 4.17. Define $\omega = \mathbf{e}^{j2\pi/n}$, $\eta = \mathbf{e}^{j\pi/n}$, and denote $\omega_k^+ = \omega^k$ and $\omega_k^- = \omega_k^+ \eta$. Define the block Fourier-like matrices \mathcal{F}_\pm by

$$(\mathcal{F}_\pm)_{j,k} = \frac{1}{\sqrt{n}} (\omega_{j-1}^\pm)^{k-1} I_m. \quad (99)$$

Then the block-circulant matrices $X_{\pm 1}$ are diagonalized as

$$X_{\pm 1} = \mathcal{F}_{\pm 1}^H \mathcal{D}(\mathcal{S}_\pm) \mathcal{F}_{\pm 1}, \quad (100)$$

where $\mathcal{S}_{\pm 1} = \{\omega_{k-1}^\pm I_m\}$.

Proof. Denote by $\mathbf{F}_{\pm 1} \in \mathbb{C}^{n \times n}$ the matrices

$$(\mathbf{F}_{\pm 1})_{j,k} = \frac{1}{\sqrt{n}} (\omega_{j-1}^\pm)^{k-1}.$$

Then $\mathbf{F}_{-1} = \mathbf{F}_{+1} \text{diag}(\mathbf{e}^{j\pi k/n}) = \mathbf{F} \Lambda_{-1}$. It is evident that $\mathcal{F}_{\pm 1} = \mathbf{F}_{\pm 1} \otimes I_m$. Similarly, let $S_{\pm 1}$ denote the diagonal matrices

$$(S_{\pm 1})_{i,i} = \text{diag}(\omega_{i-1}^\pm).$$

Then $\mathcal{D}(\mathcal{S}_{\pm 1}) = S_{\pm 1} \otimes I_m$. From previous developments, $Z_{\pm 1} = F_{\pm 1}^H S_{\pm 1} F_{\pm 1}$, and therefore $X_{\pm 1} = \mathcal{F}_{\pm 1}^{-1} \mathcal{D}(\mathcal{S}_{\pm 1}) \mathcal{F}_{\pm 1}$. \square

In light of these developments, a two-level Toeplitz matrix may be transformed into a generalized block-Cauchy matrix.

Theorem 4.5. Let $T = [A_{i-j}]$ be (m, n) two-level Toeplitz with blocks A_{i-j} , define G and H as in (98). Let $U = \mathcal{F}_{-1} G$ and $V^T = H^T \mathcal{F}_{+1}^H$. Then $\Phi = \mathcal{F}_{-1} T \mathcal{F}_{+1}^H$ is a generalized block-Cauchy matrix.

Proof.

$$\begin{aligned} U^T V &= \mathcal{F}_{-1} G H^T \mathcal{F}_{+1}^H = \mathcal{F}_{-1} (X_{-1} T - T X_{+1}) \mathcal{F}_{+1}^H \\ &= \mathcal{D}(\mathcal{S}_{-1}) \mathcal{F}_{-1} T \mathcal{F}_{+1}^H - \mathcal{F}_{-1} T \mathcal{F}_{+1}^H \mathcal{D}(\mathcal{S}_{+1}) = \mathcal{D}(\mathcal{S}_{-1}) \Phi - \Phi \mathcal{D}(\mathcal{S}_{+1}). \end{aligned}$$

Since the blocks of $\mathcal{D}(\mathcal{S}_\pm)$ are all scaled versions of the identity matrix, the necessary commutativity and inversion conditions of Definition 4.3 are met. By Proposition 4.15, then, Φ is generalized block-Cauchy. \square

As a consequence of Theorem 4.5, instead of solving a two-level Toeplitz system $Tx = y$, one can instead solve the system

$$Tx = \mathcal{F}_{-1}^H \mathcal{F}_{-1} T \mathcal{F}_{+1}^H \mathcal{F}_{+1} x = \mathcal{F}_{-1}^H \Phi(\mathcal{F}_{+1} x) = y,$$

or equivalently

$$\Phi(\mathcal{F}_{+1} x) = (\mathcal{F}_{-1} y).$$

4.3.4.3 Further transformation

The next natural question is whether the second level of structure may be simultaneously exploited. The following theorem gives an affirmative answer, and states that a two-level Toeplitz matrix can be transformed with 2-D Fourier-like operators to yield a two-level Loewner structure.

Theorem 4.6. *Let T be an (m, n) two-level Toeplitz matrix with blocks $A_{i-j} = [a_{i-j, k-\ell}]$.*

Define the Fourier-like matrices

$$\begin{aligned} (\mathbf{F}_{\pm 1}^{(n)})_{j,k} &= \frac{1}{\sqrt{n}} (\omega_{j-1}^{\pm})^{k-1} \quad \text{and} \\ (\mathbf{F}_{\pm 1}^{(m)})_{j,k} &= \frac{1}{\sqrt{m}} (\phi_{j-1}^{\pm})^{k-1}, \end{aligned}$$

where $\omega = \mathbf{e}^{j2\pi/n}$, $\eta = \mathbf{e}^{j\pi/n}$, $\phi = \mathbf{e}^{j2\pi/m}$, $\nu = \mathbf{e}^{j\pi/m}$, $\omega_k^+ = \omega^k$, $\omega_k^- = \omega_k^+ \eta$, $\phi_k^+ = \phi^k$, and $\phi_k^- = \phi_k^+ \nu$. Define four Fourier-like transformations

$$[\Theta_{00}]_{i,j} = \frac{1}{mn} \sum_{k=1-n}^{n-1} \sum_{s=1-m}^{m-1} a_{k,s} (\omega_{i-1}^-)^k (\phi_{j-1}^-)^s, \quad (101)$$

$$[\Theta_{01}]_{i,j} = -\frac{1}{mn} \sum_{k=1-n}^{n-1} \sum_{s=1-m}^{m-1} a_{k,s} (\omega_{i-1}^-)^k (\phi_{j-1}^+)^s, \quad (102)$$

$$[\Theta_{10}]_{i,j} = -\frac{1}{mn} \sum_{k=1-n}^{n-1} \sum_{s=1-m}^{m-1} a_{k,s} (\omega_{i-1}^+)^k (\phi_{j-1}^-)^s, \quad (103)$$

$$[\nu_{11}]_{i,j} = \frac{1}{mn} \sum_{k=1-n}^{n-1} \sum_{s=1-m}^{m-1} a_{k,s} (\omega_{i-1}^+)^k (\phi_{j-1}^+)^s. \quad (104)$$

The matrix $\tilde{\Phi} = (\mathbf{F}_{-1}^{(n)} \otimes \mathbf{F}_{-1}^{(m)})^H T(\mathbf{F}_{+1}^{(n)} \otimes \mathbf{F}_{+1}^{(m)})$ can be expressed as $\tilde{\Phi} = L\Delta$, where Δ is an $mn \times mn$ diagonal matrix with main diagonal given by

$$\Delta = \begin{bmatrix} \omega_0^+ \\ \vdots \\ \omega_{n-1}^+ \end{bmatrix} \otimes \begin{bmatrix} \phi_0^+ \\ \vdots \\ \phi_{m-1}^+ \end{bmatrix}$$

and L is a **block Loewner matrix** with blocks

$$\tilde{\Phi} = \left[\frac{J_i - K_j}{\omega_{i-1}^- - \omega_{j-1}^+} \right]_{i,j=1}^n, \quad (105)$$

where the matrices J_i and K_j are themselves Loewner matrices given by

$$J_i = \left[\frac{[\Theta_{00}]_{i,\ell} - [\Theta_{01}]_{i,r}}{\phi_{\ell-1}^- - \phi_{r-1}^+} \right]_{\ell,r=1}^m, \quad K_j = \left[\frac{[\Theta_{10}]_{j,\ell} - [\Theta_{11}]_{j,r}}{\phi_{\ell-1}^- - \phi_{r-1}^+} \right]_{\ell,r=1}^m. \quad (106)$$

Matrices of this form are termed **two-level Loewner matrices**.

Proof. See Appendix A. □

At first glance, Theorem 4.6 seems as if it should lend itself well to an adaption of the scalar algorithm given in [21]. However, the two-level Loewner structure that results from transforming a two-level Toeplitz matrix with Fourier-like operators does not result in the same type of compressible structure as arises in the scalar case. Specifically, the two-level nature of the Loewner structure results in high-rank blocks around the main diagonal of each block diagonal.

Figure 17 illustrates this behavior for a two-level Toeplitz matrix with $m = 512$ and $n = 4$ whose coefficients were drawn from a standard normal distribution. To generate the figure, a progressively refined dyadic partitioning of the matrix was performed, starting with the full matrix and ending with 4×4 blocks. For each partitioning, the rank of each block was computed and the ratio of the rank to the matrix dimension – a measure of the “compressibility” of a given block – was recorded. The values in the figure indicate the minimum ratio for each individual entry in the matrix, and serve to illustrate the hierarchical

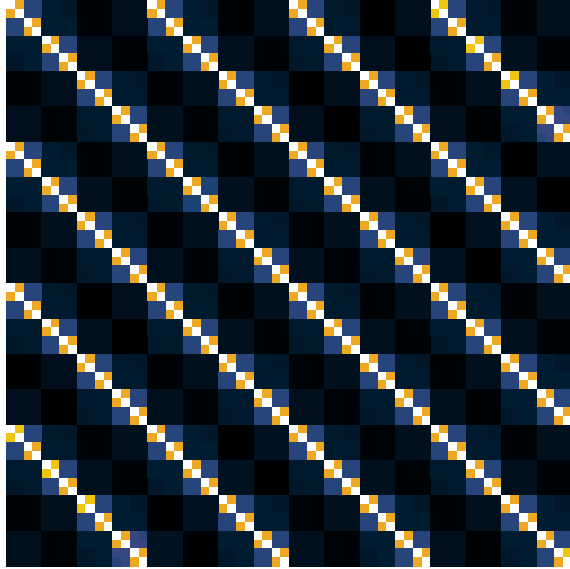


Figure 17: Map of the lowest information compression rates for each entry in a transformed two-level Toeplitz matrix. Brighter entries indicate localized areas where the information in the matrix is concentrated.

partitioning of the matrix that would result in optimal compression of the matrix. Since there are many “secondary diagonals” that also have high-rank blocks, the resulting two-level Loewner form cannot be efficiently compressed with the SSS representation of [22]. Therefore, a hierarchical approach to Toeplitz inversion that has characterized more recent scalar algorithms does not seem to extend easily to the two-level case.

Scalar generalized Cauchy matrices, and Loewner matrices in particular, have a strong connection with tangential interpolation. Correspondingly, two-level Loewner matrices can be connected to bivariate interpolation problems using manipulations similar to those of Section 2.2.2.4. Unfortunately, multivariate interpolation problems are notoriously difficult to solve, as the uniqueness and even existence of solutions may not be guaranteed. While tangential-interpolation problems are different from normal interpolation problems, many of the same difficulties arise and no established framework in the vein of [108] is available. The transformation into a two-level Loewner structure is thus interesting in that it mirrors developments used for scalar Toeplitz inversion algorithms, but does not appear to be particularly useful in practice. It neither yields structures with compressible off-diagonal

blocks nor produces interpolation problems that can be solved efficiently.

CHAPTER V: CONCLUSIONS

This thesis presented several algorithms for the efficient solution of scalar and multi-level structured linear systems. Most of the algorithms are based on the concept of matrix displacement, which allows a structured matrix to be multiplied with a considerably reduced number of operations. Displacement was also used to provide decompositions of the inverses structured matrices into sums of structured products, collectively termed structured-sum decompositions. Once these SSDs were established for a given matrix structure, they prescribed a method of applying a structured inverse (and therefore solving a linear system) in an efficient manner.

5.1 Summary of results

Tikhonov-regularized least-squares solutions to Toeplitz systems:

The first contribution is an algorithm to solve Toeplitz-structured regularization problems in $O(N \log^2 N)$ operations, where N is the total number of free parameters in the system. The algorithm translates the original linear-algebraic system into a tangential-interpolation problem, and is based on the “extension-and-transformation” approach of [53]. By using displacement operators, it was shown that the inverse matrices that describe these problems can be expressed with an SSD that requires $O(n \log n)$ to apply if the generators are computed.

Non-uniform resampling of digital signals:

The second contribution is a pair of superfast algorithms for resampling digital signals between uniform and non-uniform grids. These algorithms make use of the FMM and FFT to perform structured matrix multiplications efficiently. For the recovery of uniform samples from non-uniform samples, a superfast Schur recursion was derived that subdivides the calculation of the inverse generators. While presented in the context of resampling, these sinc-interpolation algorithms are actually applicable to a number of related problems

as well.

Inversion of semi-commutative two-level Toeplitz systems:

The next contribution is a method of extending scalar algorithms to multi-level Toeplitz matrices with commutative structures in one or more levels. These SCTLT matrices can effectively be treated as scalar Toeplitz matrices, with scalars drawn from a commutative ring rather than a field. Using this development, superfast algorithms were given to determine the inverse generators of SCTLT matrices containing either triangularity or f -circulant structure in one or more of their levels. The inverse generators were then used in multi-level extensions of the Gohberg-Semencul SSDs to apply the inverse of these special classes of two-level Toeplitz matrices in $O(n \log n)$ operations.

One-level SSDs of two-level Toeplitz inverses:

Using notions of multi-level structure and persymmetry properties, the penultimate contribution is a generalization of the Gohberg-Heinig formula to an infinite number of SSDs that exploit a single level of Toeplitz structure in two-level Toeplitz matrices. These new SSDs are analogous to the results surveyed in Section 2.1, which show that the scalar Toeplitz inverse can be expressed in SSDs constructed from any linearly independent set of vectors from the subspace spanned by the inverse generators.

The generalized Gohberg-Heinig formulas presented are indeed SSDs constructed from linearly independent block column vectors that are linear combinations of the block inverse generators. There are certain restrictions on these linear combinations to ensure that the resulting block column vectors are both linearly independent and form a valid generator pair. These results allow the two-level Toeplitz inverse to be framed in much broader and more universal terms, and make a stronger connection to scalar results than was previously available.

Properties of two-level Toeplitz inverses:

Finally, the last contribution of the work is a set of results on the properties of the two-level Toeplitz inverse and an analysis of the difficulties that arise in attempting to extend scalar

results to the multi-level case. First, results on two-level displacement defined through Kronecker products show that it is difficult to formulate a useful definition of multi-level displacement rank. Second, using the known persymmetry properties of the two-level inverse, equivalences were shown among obvious potential candidates for two-level inverse SSDs. While the existence of such SSDs is not proven, it is shown that if a certain type of SSD is found to exist it will immediately imply the existence of several equivalent forms.

Next, using similar persymmetry arguments, the problem of determining a set of inverse generators for the two-level inverse were explored. The principal objective was to make some statements regarding a set of possible generators that accurately represent the number of free parameters in the matrix while retaining definitions similar to the scalar case. Specifically, it was shown that this set conveyed the proper number of degrees of freedom, was connected to terms arising in the two-level displacement of Toeplitz matrices, and shared similar definitions to the scalar generators used in the original Gohberg-Semencul SSD.

Finally, it was shown that two-level Toeplitz matrices may be transformed with Fourier-like operators to yield two-level Loewner matrices. This result is an analog of known results on transformations of Toeplitz matrices that are employed in compressive inversion algorithms. However, the transformed two-level matrices do not share the same properties as their scalar counterparts, exhibiting many blocks with large rank outside of the main diagonal. For this reason, the compressive scalar inversion algorithms are difficult to adapt to the two-level case.

5.2 Future directions

To conclude the thesis, the following is a list of potential future directions to be explored regarding each of the major contributions of the work.

Tikhonov-regularized least-squares solutions to Toeplitz systems

The experiments used to demonstrate the performance of the algorithm made use of synthetic data sets and contrived problems. It would be interesting to apply the algorithm to a problem that:

1. involves a large data set for which alternative direct-inversion schemes would be impractical;
2. requires regularization due to the conditioning of the linear system to be solved; and
3. contains Toeplitz structure in its system matrix and in the desired regularizers.

Potential candidates are applications involving the recovery of large 1-D spatial signals from non-uniform Fourier measurements. Such an application has the potential to meet the above criteria, and given how often non-uniform Fourier measurements appear in signal-processing problems it is likely that the algorithm could be put to practical use.

Non-uniform resampling of digital signals

Similarly, the experiments demonstrating the use of non-uniform resampling with structured matrix operations used simulated scenarios as the Tikhonov problem. There are real applications of interest – such as level-crossing ADCs and SAR imaging – where these algorithms might be put to great use. A potential future direction, then, is to test the performance of these algorithms on real-world data.

Multi-level results

The most obvious future direction for the results of Chapter 4 is in furthering attempts to determine the structure of the two-level Toeplitz inverse. The three main contributions in this area may function jointly toward this goal. The specialized SCTLT inversion formulas are simply special cases of two-level inverses, and therefore any valid SSD of the two-level inverse must collapse to these forms when the two-level Toeplitz matrix to be inverted is also semi-commutative. Similarly, the one-level SSDs that serve as extensions of the Gohberg-Heinig formula must be equivalent to more efficient SSDs for two-level matrices. By studying the properties of the block generators that form these decompositions, it might

be possible to determine the proper SSDs and generators of two-level inverses. Finally, since there are many ways to “attack” the problem of describing the two-level inverse, the final contributions serve as a starting point to considerably narrow the search for generators and SSDs. The results of Chapter 4 may then be seen as a foundation for future progress on the two-level Toeplitz inverse problem.

APPENDIX A

PROOFS

A.1 Chapter 1

Proof of Proposition 1.3. For Sylvester displacement, consider the $((j-1)m+i)^{th}$ entry of each side:

$$\begin{aligned} e_{i+(j-1)m}^T \mathbf{vec}(\nabla_{A,B}(C)) &= (AC - CB)_{i,j} = e_i^T ACe_j - e_i^T CBe_j \\ &= \left(\sum_{k=1}^m A_{i,k} C_{k,j} \right) - \left(\sum_{k=1}^n C_{i,k} B_{k,j} \right) \end{aligned}$$

and

$$\begin{aligned} e_{i+(j-1)m}^T \Pi_{A,B}(\mathbf{vec}(C)) &= e_{i+(j-1)m}^T (\mathbf{I}_n \otimes A - B^T \otimes \mathbf{I}_m) \mathbf{vec}(C) \\ &= (e_j^T \otimes e_i^T) (\mathbf{I}_n \otimes A - B^T \otimes \mathbf{I}_m) \mathbf{vec}(C) \\ &= (e_j^T \otimes (e_i^T A) - (e_j^T B^T) \otimes e_i^T) \mathbf{vec}(C) \\ &= \left(\sum_{k=1}^m A_{i,k} C_{k,j} \right) - \left(\sum_{k=1}^n C_{i,k} B_{k,j} \right). \end{aligned}$$

Next, the $(i+(j-1)m)^{th}$ entry of the Stein displacement equation is

$$\begin{aligned} e_{i+(j-1)m}^T \mathbf{vec}(\Delta_{A,B}(C)) &= (C - ACB)_{i,j} = C_{i,j} - e_i^T ACBe_j \\ &= C_{i,j} - \sum_{k=1}^m \sum_{\ell=1}^n A_{i,k} C_{k,\ell} B_{\ell,j} \end{aligned}$$

and

$$\begin{aligned} e_{i+(j-1)m}^T \Theta_{A,B}(\mathbf{vec}(C)) &= e_{i+(j-1)m}^T (\mathbf{I}_{mn} - B^T \otimes A) \mathbf{vec}(C) \\ &= (e_{i+(j-1)m}^T \mathbf{vec}(C) - (e_j^T \otimes e_i^T) (B^T \otimes A) \mathbf{vec}(C)) \\ &= (\mathbf{vec}(C))_{i+(j-1)m} - ((e_j^T B^T) \otimes (e_i^T \otimes A)) \mathbf{vec}(C) \\ &= C_{i,j} - \sum_{k=1}^m \sum_{\ell=1}^n A_{i,k} (B^T)_{j,\ell} (\mathbf{vec}(C))_{(k-1)n+\ell} \end{aligned}$$

$$= C_{i,j} - \sum_{k=1}^m \sum_{\ell=1}^n A_{i,k} C_{k,\ell} B_{\ell,j}.$$

□

Proof of Proposition 1.5. The first point follows directly from the definition of persymmetry. The second point may be shown by simply invoking the two transposition operations:

$$(A^T)^P = J_m A J_n = (J_n^T A^T J_m^T)^T = (J_n A^T J_m)^T = (A^P)^T.$$

Regarding the third point, for A nonsingular it follows that

$$(A^{-1})^P = J_n (A^{-1})^T J_n = J_n (A^T)^{-1} J_n = (J_n A^T J_n)^{-1} = (A^P)^{-1}.$$

Therefore, without ambiguity, $A^{-P} = (A^{-1})^P = (A^P)^{-1}$. Finally, the fourth point is proven as follows:

$$(AB)^P = J_p (AB)^T J_m = J_p B^T J_n J_n A^T J_m = B^P A^P.$$

□

Proof of Proposition 1.7. Beginning with the first item, let $B = AC$ for $C \in \mathbb{R}^{n \times n}$ persymmetric. Then $AB^P = AC^P A^P = ACA^P = BA^P$. For the reverse direction, let $C = A^{-1}B$. Then if $AB^P = BA^P$, it follows that

$$A^{-1}BA^P = A^{-1}AB^P$$

$$CA^P = B^P$$

$$C = B^P A^{-P} = (A^{-1}B)^P = C^P.$$

Letting $\tilde{A} = A^P$, the second item follows from the first: $\tilde{A}B = B^P \tilde{A}$ if and only if $B^P = \tilde{A}C$, or $B = C^P \tilde{A}^P = CA$. □

A.2 Chapter 2

Proof of Proposition 2.4. It is not possible to directly appeal to Fact 1.2 to determine the inverse displacements, as Z_0^T and Z_0 are singular. However, from their definitions, the following are true:

$$\begin{aligned}\Delta_{Z_0^T, Z_0}(B) &= \Delta_{Z_1^T, Z_0}(B) - e_n e_1^T B Z_0 \quad \text{and} \\ \Delta_{Z_0, Z_0^T}(B) &= \Delta_{Z_1, Z_0^T}(B) - e_1 e_n^T B Z_0^T.\end{aligned}$$

Since $(h_{[0]})_1 = 0$ and $(g_{[0]})_n = 0$, the following identities hold:

$$\begin{aligned}h_{[0]}^P Z_0^T Z_1 &= h_{[0]}^P (Z_1^T - e_n e_1^T) Z_1 = h_{[0]}^P - h_{[0]}^P e_n e_1^T = h_{[0]}^P \\ g_{[0]}^P Z_0 Z_1^T &= g_{[0]}^P (Z_1 - e_1 e_n^T) Z_1^T = g_{[0]}^P - g_{[0]}^P e_1 e_n^T = g_{[0]}^P.\end{aligned}$$

It then follows from Fact 1.2 and Proposition 2.3 that

$$\begin{aligned}\Delta_{Z_1^T, Z_0}(B) &= Z_1^T B (e_1 h_{[0]}^P Z_0^T - h_{[0]} e_1^T + T e_1 e_1^T) Z_1 B \\ &= Z_1^T B e_1 h_{[0]}^P B - Z_1^T B h_{[0]} e_n^T B + e_n e_n^T B \\ &= Z_1^T u v_{[0]}^P - Z_1^T v_{[0]} u^P + e_n u^P \\ &= (Z_0^T + e_n e_1^T) u v_{[0]}^P - (Z_0^T + e_n e_1^T) v_{[0]} u^P + e_n u^P \\ &= \dot{u} v_{[0]}^P - (Z_0^T v_{[0]} - e_n) u^P + e_n (u_1 v_{[0]}^P - (v_{[0]})_1 u^P) \\ &= \dot{u} v_{[0]}^P - \dot{v}_{[0]} u^P + e_n (u_1 v_{[0]}^P - (v_{[0]})_1 u^P).\end{aligned}$$

Modifying the displacement so that Z_1^T is changed to Z_0^T yields

$$\Delta_{Z_0^T, Z_0}(B) = \dot{u} v_{[0]}^P - \dot{v}_{[0]} u^P + e_n (u_1 v_{[0]}^P - (v_{[0]})_1 u^P - e_1^T B Z_0).$$

From Proposition 2.2,

$$Z_0 B e_n = B Z_0 e_n + u v_{[0]}^P e_n - v_{[0]} u^P e_n = (v_{[0]})_1 u - u_1 v_{[0]},$$

and therefore $e_1^T B Z_0 = (Z_0 B e_n)^P = (v_{[0]})_1 u^P - u_1 v_{[0]}^P$. From this, it is evident that

$$\Delta_{Z_0^T, Z_0}(B) = \dot{u} v_{[0]}^P - \dot{v}_{[0]} u^P.$$

Since $v_{[\alpha]} = v_{[0]} + \alpha u$, it follows that

$$\begin{aligned}\Delta_{Z_0^T, Z_0}(B) &= \dot{u}v_{[0]}^P - \dot{v}_{[0]}u^P = \dot{u}(v_{[\alpha]} - \alpha u)^P - (Z_0^T(v_{[\alpha]} - \alpha u) - e_n)u^P \\ &= \dot{u}v_{[\alpha]}^P - \alpha \dot{u}u^P - (Z_0^T v_{[\alpha]} - e_n)u^P + \alpha(Z_0^T u)u^P \\ &= \dot{u}v_{[\alpha]}^P - \alpha \dot{u}u^P - \dot{v}_{[\alpha]}u^P + \alpha \dot{u}u^P = \dot{u}v_{[\alpha]}^P - \dot{v}_{[\alpha]}u^P.\end{aligned}$$

A similar proof follows for the second equality. \square

Proof of Proposition 2.5. Let A be lower-triangular Toeplitz; then $h_{[\alpha]} = \alpha e_1$ and thus $\nabla_{Z_0, Z_0}(A) = \mathbf{0}$.

Next, let $A = [a_{i,j}]$ and suppose $\nabla_{Z_0, Z_0}(A) = \mathbf{0}$. By definition, the displacement of A is

$$\nabla_{Z_0, Z_0}(A) = \begin{bmatrix} 0 & \cdots & 0 & 0 \\ a_{1,1} & \cdots & a_{1,n-1} & a_{1,n} \\ \vdots & \ddots & \vdots & \vdots \\ a_{n-1,1} & \cdots & a_{n-1,n-1} & a_{n,n} \end{bmatrix} - \begin{bmatrix} a_{1,2} & \cdots & a_{1,n} & 0 \\ a_{2,2} & \cdots & a_{2,n} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ a_{n,2} & \cdots & a_{n,n} & 0 \end{bmatrix}.$$

Since $\nabla_{Z_0, Z_0}(A) = \mathbf{0}$, the first row and last column of the displacement imply

$$a_{1,2} = \cdots = a_{1,n} = 0 \quad \text{and} \quad a_{1,n} = \cdots = a_{n-1,n} = 0,$$

respectively. For $i = 2, \dots, n$ and $j = 1, \dots, n-1$, $\nabla_{Z_0, Z_0}(A) = \mathbf{0}$ implies $a_{i-1,j} = a_{i,j+1}$, and thus $a_{i,j} = a_{i+1,j+1}$ for $1 \leq i, j \leq n-1$. By definition, then, A is lower-triangular Toeplitz.

It follows that $\nabla_{Z_0, Z_0}(A^T) = \mathbf{0}$ if and only if A^T is lower-triangular Toeplitz, which implies $\nabla_{Z_0^T, Z_0^T}(A) = \mathbf{0}$ if and only if A is upper-triangular Toeplitz. \square

Proof of Proposition 2.7. From Proposition 1.2, it suffices to show that the displacement operators $\Delta_{Z_1^T, Z_0}(\cdot)$ and $\Delta_{Z_0, Z_1^T}(\cdot)$ are nonsingular. Writing $A = [a_{i,j}]$, from (8),

$$\begin{aligned}\Delta_{Z_0, Z_1^T}(A) &= A - Z_0 A Z_1^T \\ &= \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} - \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{1,n} & a_{1,1} & \cdots & a_{1,(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{(n-1),n} & a_{(n-1),2} & \cdots & a_{(n-1),(n-1)} \end{bmatrix}.\end{aligned}$$

If $\Delta_{Z_0, Z_1^T}(A) = \mathbf{0}$, then $a_{i,1} = 0$ for $1 \leq i \leq n$ from the first row. This equation also implies that the coefficients along the $-k$ diagonal for $0 \leq k < n$ must all be zero, which implies that the last column is zero (and therefore the first column is zero). As a consequence, the coefficients along *all* of the diagonals must be zero, and therefore $A = \mathbf{0}$.

Similar logic applies for $\Delta_{Z_1^T, Z_0}(A)$, and so $\Delta_{Z_1^T, Z_0}(\cdot)$ and $\Delta_{Z_0, Z_1^T}(\cdot)$ are nonsingular. Since Z_1 and Z_1^T are nonsingular, by Proposition 1.2, $\nabla_{Z_1, Z_0}(\cdot)$ and $\nabla_{Z_0, Z_1}(\cdot)$ are nonsingular. \square

Proof of Lemma 2.1. It is immediately evident that

$$GH^T = \sum_{j=1}^r Ge_j e_j^T H = \sum_{i=1}^r g_j h_j^T.$$

Therefore,

$$\sum_{i=0}^{n-1} (Z_0^T)^i GH^T Z_0^i = \sum_{i=0}^{n-1} (Z_0^T)^i \left(\sum_{j=1}^r g_j h_j^T \right) Z_0^i = \sum_{j=1}^r \sum_{i=0}^{n-1} (Z_0^T)^i g_j h_j^T Z_0^i,$$

and similarly

$$\sum_{i=0}^{n-1} Z_0^i GH^T (Z_0^T)^i = \sum_{j=1}^r \sum_{i=0}^{n-1} Z_0^i g_j h_j^T (Z_0^T)^i.$$

For any vectors $x, y \in \mathbb{C}^{n \times 1}$, direct evaluation shows

$$\sum_{i=0}^{n-1} (Z_0^T)^i x y^T Z_0^i = \begin{bmatrix} x_n & \cdots & x_1 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & x_n \end{bmatrix} \begin{bmatrix} y_n & \cdots & 0 \\ \vdots & \ddots & \vdots \\ y_1 & \cdots & y_n \end{bmatrix}.$$

Therefore, (27) holds.

Similarly, a direct evaluation shows

$$\sum_{i=0}^{n-1} Z_0^i x y^T (Z_0^T)^i = \begin{bmatrix} x_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ x_n & \cdots & x_1 \end{bmatrix} \begin{bmatrix} y_1 & \cdots & y_n \\ \vdots & \ddots & \vdots \\ 0 & \cdots & y_1 \end{bmatrix}.$$

Therefore, (28) holds. \square

Proof of Theorem 2.1. It is easiest to prove the points in succession.

1. Suppose $\Delta_{Z_0^T, Z_0}(B) = GH^T$; from Theorem 1.1 and Lemma 2.1,

$$B = \sum_{i=1}^r U(g_i) \cdot L(h_i).$$

Conversely, if $B = \sum_{i=1}^r U(g_i) \cdot L(h_i)$ then from Lemma 2.1

$$B = \sum_{i=0}^{n-1} (Z_0^T)^i GH^T Z_0^i$$

and thus

$$\begin{aligned} \Delta_{Z_0^T, Z_0}(B) &= \sum_{i=0}^{n-1} (Z_0^T)^i GH^T Z_0^i - Z_0^T \left(\sum_{i=0}^{n-1} (Z_0^T)^i GH^T Z_0^i \right) Z_0 \\ &= GH^T + (Z_0^T)^n GH^T Z_0^n = GH^T. \end{aligned}$$

2. Since B is persymmetric, the following identity holds:

$$\nabla_{Z_0, Z_0^T}(B) = B - Z_0 B Z_0^T = B^P - Z_0^P B^P (Z_0^T)^P = (B - Z_0^T B Z_0)^P = (\nabla_{Z_0^T, Z_0}(B))^P.$$

From point 1, $B = \sum_{i=1}^r U(g_i) \cdot L(h_i)$ if and only if $\Delta_{Z_0, Z_0^T}(B) = (GH^T)^P = J_n H G^T J_n$.

3. The Sylvester displacement can be put in terms of the Stein displacement:

$$\begin{aligned} Z_1 B - B Z_0 &= Z_1 (B - Z_1^T B Z_0) = Z_1 (B - Z_0^T B Z_0 - e_n e_1^T B Z_0) \\ &= Z_1 \Delta_{Z_0^T, Z_0}(B) - e_1 e_1^T B Z_0 = Z_1 \Delta_{Z_0^T, Z_0}(B) - e_1 \dot{w}^P. \end{aligned}$$

From point 1, $B = \sum_{i=1}^r U(g_i) \cdot L(h_i)$ if and only if $\nabla_{Z_1, Z_0}(B) = Z_1 GH^T - e_1 \dot{w}^P$.

4. Manipulating the displacement operator yields

$$\begin{aligned} Z_0 B - B Z_1 &= (Z_0 B Z_1^T - B) Z_1 = -(B - Z_0 B Z_1^T) Z_1 \\ &= -(B - Z_0 B Z_0^T - Z_0 B e_n e_1^T) Z_1 \\ &= \dot{w} e_n^T - \Delta_{Z_0, Z_0^T}(B) Z_1. \end{aligned}$$

From point 2, $B = \sum_{i=1}^r U(g_i) \cdot L(h_i)$ if and only if $\nabla_{Z_0, Z_1}(B) = \dot{w} e_n^T - J_n H G^T J_n Z_1$.

5. Manipulating the displacement operator yields

$$\begin{aligned}
Z_1^T B - BZ_0 &= Z_1^T (B - Z_1 BZ_0^T) = Z_1^T (B - Z_0 BZ_0^T - e_1 e_n^T BZ_0^T) \\
&= Z_1^T \Delta_{Z_0, Z_0^T}(B) - e_n \dot{u}^P \\
&= Z_1^T J_n H G^T J_n - e_n \dot{u}^P.
\end{aligned}$$

From point 2, $B = \sum_{i=1}^r U(g_i) \cdot L(h_i)$ if and only if $\nabla_{Z_1^T, Z_0^T}(B) = Z_1^T J_n H G^T J_n - e_n \dot{u}^P$.

6. Manipulating the displacement operator yields

$$\begin{aligned}
Z_0^T B - BZ_1^T &= (Z_0^T BZ_1 - B)Z_1^T = (Z_0^T BZ_0 - B + Z_0^T B e_1 e_n^T)Z_1^T \\
&= \dot{u} e_1^T - \Delta_{Z_0^T, Z_0}(B)Z_1^T.
\end{aligned}$$

From point 1, $B = \sum_{i=1}^r U(g_i) \cdot L(h_i)$ if and only if $\nabla_{Z_0^T, Z_1^T}(B) = \dot{u} e_1^T - G H^T Z_1^T$.

□

Proof of Theorem 2.2. Several relations can be proven easily with the Stein displacement results. Using Proposition 2.4,

$$\begin{aligned}
B e_1 &= Z_0 B Z_0^T e_1 + \dot{w} x^P e_1 - \dot{x} w^P e_1 \\
u &= x_n \dot{w} - w_n \dot{x} = x_n \dot{w} - u_1 \dot{x}. \\
B e_n &= Z_0^T B Z_0 e_n + \dot{u} v^P e_n - \dot{v} u^P e_n \\
w &= v_1 \dot{u} - u_1 \dot{v}.
\end{aligned}$$

Similarly, using Proposition 2.2,

$$\begin{aligned}
Z_0 B e_n &= B Z_0 e_n + u v^P e_n - v u^P e_n \\
\dot{w} &= v_1 u - u_1 v. \\
Z_0^T B e_1 &= B Z_0^T e_1 + w x^P e_1 - x w^P e_1 \\
\dot{u} &= x_n w - w_n x.
\end{aligned}$$

These equalities imply the following:

$$\begin{aligned}
 u &= u_1 v_1^{-1} v + v_1^{-1} \dot{w} \\
 v &= v_1 u_1^{-1} u - u_1^{-1} \dot{w} \\
 w &= x_n^{-1} \dot{u} + u_1 x_n^{-1} x \\
 x &= -u_1^{-1} \dot{u} + x_n u_1^{-1} w.
 \end{aligned}$$

Equating the two identities for u yields

$$\begin{aligned}
 x_n \dot{w} - u_1 \dot{x} &= u_1 v_1^{-1} v + v_1^{-1} \dot{u} \\
 u_1 v_1^{-1} v &= (x_n - v_1^{-1}) \dot{w} - u_1 \dot{x} \\
 v &= -\lambda^{-1} \dot{w} - v_1 \dot{x}.
 \end{aligned}$$

Equating the two identities for w yields

$$\begin{aligned}
 v_1 \dot{u} - u_1 \dot{v} &= x_n^{-1} \dot{u} + u_1 x_n^{-1} x \\
 u_1 x_n^{-1} x &= (v_1 - x_n^{-1}) \dot{u} - u_1 \dot{v} \\
 x &= -\lambda^{-1} \dot{u} - x_n \dot{v}.
 \end{aligned}$$

It is then possible to prove the remaining four identities.

$$\begin{aligned}
 u &= x_n \dot{w} - u_1 \dot{x} = x_n (v_1 u - u_1 v) - u_1 \dot{x} \\
 &= -x_n u_1 (1 - x_n v_1)^{-1} v - u_1 (1 - x_n v_1)^{-1} \dot{x} = -\lambda x_n v - \lambda \dot{x} \\
 v &= -\lambda^{-1} \dot{w} - v_1 \dot{x} = -\lambda^{-1} x_n^{-1} (u + u_1 \dot{x}) - v_1 \dot{x} \\
 &= -(\lambda x_n)^{-1} u - x_n^{-1} \dot{x} \\
 w &= v_1 \dot{u} - u_1 \dot{v} = v_1 (-\lambda x - \lambda x_n \dot{v}) - u_1 \dot{v} \\
 &= -(\lambda v_1 x_n + u_1) \dot{v} - \lambda v_1 x = -\lambda \dot{v} - \lambda v_1 x \\
 x &= -\lambda^{-1} \dot{u} - x_n \dot{v} = -\lambda^{-1} v_1^{-1} (w + u_1 \dot{v}) - x_n \dot{v} \\
 &= -(u_1 \lambda^{-1} v_1^{-1} + x_n) \dot{v} - (\lambda v_1)^{-1} w = -v_1^{-1} \dot{v} - (\lambda v_1)^{-1} w.
 \end{aligned}$$

□

Proof of Theorem 2.3. From Corollary 2.3, it suffices for each point to show that the Stein displacement of B is equal to the term inside $\mathcal{G}(\cdot)$.

1. This is the Barnett formula, as given in [56], [91], [8] and [79], and it follows directly from Proposition 2.4; that is,

$$\Delta_{z_0^T, z_0}(B) = \begin{bmatrix} \dot{u} & \dot{v} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v^P \\ u^P \end{bmatrix}.$$

Corollary 2.3 completes the proof.

2. This is the original Gohberg-Semencul formula, given in [48]. From Theorem 2.2, the following identities hold:

$$\begin{bmatrix} \dot{u} & \dot{v} \end{bmatrix} \begin{bmatrix} 1 & v_1 \\ 0 & -u_1 \end{bmatrix} = \begin{bmatrix} \dot{u} & w \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -u_1 & v_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v^P \\ u^P \end{bmatrix} = \begin{bmatrix} \dot{w}^P \\ u^P \end{bmatrix}.$$

Letting $A = \begin{bmatrix} 1 & v_1 \\ 0 & -u_1 \end{bmatrix}$, it follows that A is nonsingular with $\det(A) = -u_1$, and therefore from the proof of Corollary 2.4 it follows that

$$\begin{aligned} \Delta_{z_0^T, z_0}(B) &= \begin{bmatrix} \dot{u} & \dot{v} \end{bmatrix} A A^{-1} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} A^{-P} A^P \begin{bmatrix} v^P \\ u^P \end{bmatrix} \\ &= \begin{bmatrix} \dot{u} & w \end{bmatrix} \begin{bmatrix} -u_1^{-1} & 0 \\ 0 & u_1^{-1} \end{bmatrix} \begin{bmatrix} \dot{w}^P \\ u^P \end{bmatrix}. \end{aligned}$$

Corollary 2.3 completes the proof.

3. From Theorem 2.2, the following identities hold:

$$\begin{bmatrix} \dot{u} & \dot{v} \end{bmatrix} \begin{bmatrix} 1 & -\lambda^{-1} \\ 0 & -x_n \end{bmatrix} = \begin{bmatrix} \dot{u} & x \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -x_n & -\lambda^{-1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v^P \\ u^P \end{bmatrix} = \begin{bmatrix} \dot{x}^P \\ u^P \end{bmatrix}.$$

Letting $A = \begin{bmatrix} 1 & -\lambda^{-1} \\ 0 & -x_n \end{bmatrix}$, it follows that A is nonsingular with $\det(A) = -x_n$, and therefore from the proof of Corollary 2.4

$$\begin{aligned} \Delta_{Z_0^T, Z_0}(B) &= \begin{bmatrix} \dot{u} & \dot{v} \end{bmatrix} A A^{-1} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} A^{-P} A^P \begin{bmatrix} v^P \\ u^P \end{bmatrix} \\ &= \begin{bmatrix} \dot{u} & x \end{bmatrix} \begin{bmatrix} -x_n^{-1} & 0 \\ 0 & x_n^{-1} \end{bmatrix} \begin{bmatrix} \dot{x}^P \\ u^P \end{bmatrix}. \end{aligned}$$

Corollary 2.3 completes the proof.

4. From Theorem 2.2, the following identities hold:

$$\begin{bmatrix} \dot{u} & \dot{v} \end{bmatrix} \begin{bmatrix} v_1 & 0 \\ -u_1 & 1 \end{bmatrix} = \begin{bmatrix} w & \dot{v} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 0 \\ -u_1 & v_1 \end{bmatrix} \begin{bmatrix} v^P \\ u^P \end{bmatrix} = \begin{bmatrix} v^P \\ \dot{w}^P \end{bmatrix}.$$

Letting $A = \begin{bmatrix} v_1 & 0 \\ -u_1 & 1 \end{bmatrix}$, it follows that A is nonsingular with $\det(A) = v_1$, and therefore from the proof of Corollary 2.4

$$\begin{aligned} \Delta_{Z_0^T, Z_0}(B) &= \begin{bmatrix} \dot{u} & \dot{v} \end{bmatrix} A A^{-1} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} A^{-P} A^P \begin{bmatrix} v^P \\ u^P \end{bmatrix} \\ &= \begin{bmatrix} w & \dot{v} \end{bmatrix} \begin{bmatrix} v_1^{-1} & 0 \\ 0 & -v_1^{-1} \end{bmatrix} \begin{bmatrix} v^P \\ \dot{w}^P \end{bmatrix}. \end{aligned}$$

Corollary 2.3 completes the proof.

5. From Theorem 2.2, the following identities hold:

$$\begin{bmatrix} \dot{u} & \dot{v} \end{bmatrix} \begin{bmatrix} -\lambda^{-1} & 0 \\ -x_n & 1 \end{bmatrix} = \begin{bmatrix} x & \dot{v} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 0 \\ -x_n & -\lambda^{-1} \end{bmatrix} \begin{bmatrix} v^P \\ u^P \end{bmatrix} = \begin{bmatrix} v^P \\ \dot{x}^P \end{bmatrix}.$$

Letting $A = \begin{bmatrix} 1 & 0 \\ -x_n & -\lambda^{-1} \end{bmatrix}$, it follows that A is nonsingular with $\det(A) = -\lambda^{-1}$, and

therefore from the proof of Corollary 2.4

$$\begin{aligned}\Delta_{Z_0^T, Z_0}(B) &= \begin{bmatrix} \dot{u} & \dot{v} \end{bmatrix} A A^{-1} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} A^{-P} A^P \begin{bmatrix} v^P \\ u^P \end{bmatrix} \\ &= \begin{bmatrix} x & \dot{v} \end{bmatrix} \begin{bmatrix} \lambda & 0 \\ 0 & -\lambda \end{bmatrix} \begin{bmatrix} v^P \\ \dot{x}^P \end{bmatrix}.\end{aligned}$$

Corollary 2.3 completes the proof.

6. From Theorem 2.2, the following identities hold:

$$\begin{bmatrix} \dot{u} & \dot{v} \end{bmatrix} \begin{bmatrix} v_1 & -\lambda^{-1} \\ -u_1 & -x_n \end{bmatrix} = \begin{bmatrix} w & x \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -x_n & -\lambda^{-1} \\ -u_1 & v_1 \end{bmatrix} \begin{bmatrix} v^P \\ u^P \end{bmatrix} = \begin{bmatrix} \dot{x}^P \\ \dot{w}^P \end{bmatrix}.$$

Letting $A = \begin{bmatrix} v_1 & -\lambda^{-1} \\ -u_1 & -x_n \end{bmatrix}$, it follows that A is nonsingular with $\det(A) = -1$, and therefore from the proof of Corollary 2.4

$$\begin{aligned}\Delta_{Z_0^T, Z_0}(B) &= \begin{bmatrix} \dot{u} & \dot{v} \end{bmatrix} A A^{-1} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} A^{-P} A^P \begin{bmatrix} v^P \\ u^P \end{bmatrix} \\ &= \begin{bmatrix} w & x \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}^P \\ \dot{w}^P \end{bmatrix}.\end{aligned}$$

□

Proof of Theorem 2.5. It first needs to be shown that

$$\text{rank}(A - Z_0 A Z_0^T) = \text{rank}(A^{-1} - Z_0^T A^{-1} Z_0).$$

Since A is full rank,

$$\begin{aligned}\text{rank}(A - Z_0 A Z_0^T) &= \text{rank}((\mathbf{I} - Z_0 A Z_0^T A^{-1})A) \\ &= \text{rank}(\mathbf{I} - Z_0 A Z_0^T A^{-1}).\end{aligned}$$

From Propositions 2.8.3 and 2.8.4 of [12],

$$\text{rank} \left(\begin{bmatrix} \mathbf{I} & Z_0 A \\ Z_0^T A^{-1} & \mathbf{I} \end{bmatrix} \right) = n + \text{rank}(\mathbf{I} - Z_0 A Z_0^T A^{-1}) = n + \text{rank}(\mathbf{I} - Z_0^T A^{-1} Z_0 A),$$

and therefore $\text{rank}(\mathbf{I} - Z_0 A Z_0^T A^{-1}) = \text{rank}(\mathbf{I} - Z_0^T A^{-1} Z_0 A)$. Thus

$$\begin{aligned} \text{rank}(A - Z_0 A Z_0^T) &= \text{rank}(\mathbf{I} - Z_0 A Z_0^T A^{-1}) \\ &= \text{rank}(\mathbf{I} - Z_0^T A^{-1} Z_0 A) = \text{rank}((A^{-1} - Z_0^T A^{-1} Z_0) A) \\ &= \text{rank}(A^{-1} - Z_0^T A^{-1} Z_0). \end{aligned}$$

As a result,

$$\text{rank}(\Delta(A)) = \text{rank}(A - Z_0 A Z_0^T) = r \Rightarrow \text{rank}(\Delta(B)) = \text{rank}(B - Z_0^T B Z_0).$$

Next, again assuming the size of the 0-shift matrices from context, the $2n \times 2n$ Z_0 may be partitioned as

$$Z_0 = \begin{bmatrix} Z_0 & \mathbf{0} \\ e_1 e_1^P & Z_0 \end{bmatrix},$$

where the size of the smaller component Z_0 s are determined from the partitioning. Using this expression, it follows that for any matrix C partitioned as A satisfies

$$\begin{aligned} \Delta(C) &= C - Z_0 C Z_0^T = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} - \begin{bmatrix} Z_0 & \mathbf{0} \\ e_1 e_1^P & Z_0 \end{bmatrix} \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} Z_0^T & e_n e_n^P \\ \mathbf{0} & Z_0^T \end{bmatrix} \\ &= \begin{bmatrix} \Delta(C_{11}) & * \\ * & * \end{bmatrix}. \end{aligned}$$

One can then see that the northwest corner of $\Delta(C)$ is $\Delta(C_{11})$. Thus, if $\Delta(A) = UV^T$, where $U, V \in \mathbb{C}^{2n \times r}$, it follows that $\Delta(A_{11}) = U_1 V_1^T$, where U_1 and V_1 are the submatrices formed from the first n rows of U and V . Therefore,

$$\text{rank}(A_{11} - Z_0 A_{11}^{-1} Z_0^T) = r$$

and thus

$$\text{rank}(A_{11}^{-1} - Z_0^T A_{11}^{-1} Z_0) = r.$$

Next remains the proof that the displacement rank of the Schur complement is also r .

From the blockwise matrix inversion formula, letting $B = A^{-1}$, it follows that

$$A^{-1} = \begin{bmatrix} A_{11}^{-1} + A_{11}^{-1} A_{12} S^{-1} A_{21} A_{11}^{-1} & -A_{11}^{-1} A_{12} S^{-1} \\ -S^{-1} A_{21} A_{11}^{-1} & S^{-1} \end{bmatrix}.$$

Using the same line of thought as before, for any matrix C partitioned as A^{-1} ,

$$\begin{aligned} \Delta(C) &= C - Z_0 C Z_0^T = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} - \begin{bmatrix} Z_0^T & e_n e_n^P \\ \mathbf{0} & Z_0^T \end{bmatrix} \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} Z_0 & \mathbf{0} \\ e_1 e_1^P & Z_0 \end{bmatrix} \\ &= \begin{bmatrix} * & * \\ * & \Delta(C_{22}) \end{bmatrix}. \end{aligned}$$

Since the displacement of A^{-1} has rank r , there are matrices $Y, Z \in \mathbb{C}^{2n \times r}$ such that $A^{-1} - Z_0^T A^{-1} Z_0 = YZ^T$. Then clearly $S^{-1} - Z_0^T S^{-1} Z_0 = Y_2 Z_2^T$, where Y_2 and Z_2 are formed from the last n rows of Y and Z . Therefore

$$\text{rank}(S - Z_0 S Z_0^T) = r$$

and thus

$$\text{rank}(S^{-1} - Z_0^T S^{-1} Z_0) = r,$$

which completes the proof. □

A.3 Chapter 3

Proof of Theorem 3.1. Basic manipulations give the following two identities:

$$\begin{aligned} Z_0 T - T Z_0 &= u_+ \hat{f}_1^T - e_1 \hat{u}^T. \\ Z_0 T^H - T^H Z_0 &= r \hat{e}_1^T - f_1 \hat{r}_-^T. \end{aligned}$$

Since $Z_1 = Z_0 + f_1 \hat{f}_1^T$, it follows that

$$Z_0 T - T Z_1 = (u_+ - T f_1) \hat{f}_1^T - e_1 \hat{u}^T.$$

Combining these two displacement structures yields

$$\begin{aligned} Z_0 G_T - G_T Z_1 &= Z_0 T^H T - T^H Z_0 T + T^H Z_0 T - T^H T Z_1 \\ &= (Z_0 T^H - T^H Z_0) T + T^H (Z_0 T - T Z_1) \\ &= r(\hat{e}_1^T T) - f_1(\hat{r}_-^T T) + (T^H(u_+ - T f_1)) \hat{f}_1^T - (T^H e_1) \hat{u}^T \\ &= r \hat{s}^T - t \hat{u}^T + v \hat{f}_1^T - f_1 \hat{w}^T. \end{aligned}$$

□

Proof of Theorem 3.2. Consider the displacement of G :

$$\begin{aligned} Z_0 G - G Z_1 &= \sum_{i=1}^K r_i \hat{s}_i^T - t_i \hat{u}_i^T + v_i \hat{f}_1^T - f_1 \hat{w}_i^T \\ &= \mu \hat{f}_1^T - f_1 \hat{v}^T + \sum_{i=1}^K (r_i \hat{s}_i^T - t_i \hat{u}_i^T). \end{aligned}$$

The displacement of the inverse can then be obtained directly:

$$\begin{aligned} Z_1 G^{-1} - G^{-1} Z_0 &= -G^{-1} (Z_0^{(n)} G - G Z_1^{(n)}) G^{-1} \\ &= (G^{-1} f_1)(\hat{v}^T G^{-1}) - (G^{-1} \mu)(\hat{f}_1^T G^{-1}) + \\ &\quad \sum_{i=1}^K ((G^{-1} t_i)(\hat{u}_i^T G^{-1}) - (G^{-1} r_i)(\hat{s}_i^T G^{-1})) \\ &= \alpha \hat{\beta}^T - \chi \hat{v}^T + \sum_{i=1}^K (\zeta_i \hat{\xi}_i^T - \theta_i \hat{\psi}_i^T), \end{aligned}$$

where the symmetry of G^{-1} was used in the last line. □

A.4 Chapter 4

Proof of Theorem 4.1. First, since T can be written as a matrix with coefficients that are drawn from a commutative ring, it follows that B can be written as a matrix with coefficients

drawn from the same ring. Therefore, the blocks of B are elements of \mathcal{R} , and by their definitions the components of \mathcal{U} and \mathcal{V} must be elements of \mathcal{R} .

Let the matrices of the decomposition be labeled such that the decomposition states $B = U_1L_1 + U_2L_2$. From the properties of Sylvester displacement,

$$\nabla_{X_0, X_0}(U_1L_1 + U_2L_2) = \nabla_{X_0, X_0}(U_1)L_1 + U_1\nabla_{X_0, X_0}(L_1) + \nabla_{X_0, X_0}(U_2)L_2 + U_2\nabla_{X_0, X_0}(L_2).$$

Since the matrices L_1 and L_2 are block lower-triangular Toeplitz matrices, they are in the nullspace of $\nabla_{X_0, X_0}(\cdot)$. Therefore,

$$\nabla_{X_0, X_0}(U_1L_1 + U_2L_2) = \nabla_{X_0, X_0}(U_1)L_2 + \nabla_{X_0, X_0}(U_2)L_1.$$

From Proposition 4.1,

$$\nabla_{X_0, X_0}(U_1) = \begin{bmatrix} \mathcal{V}_n & \cdots & \mathcal{V}_2 & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & -\mathcal{V}_2 \\ \vdots & \ddots & \vdots & -\vdots \\ \mathbf{0} & \cdots & \mathbf{0} & -\mathcal{V}_n \end{bmatrix} \quad \text{and} \quad \nabla_{X_0, X_0}(U_2) = \begin{bmatrix} -\mathcal{U}_n & \cdots & -\mathcal{U}_2 & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathcal{U}_2 \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathcal{U}_n \end{bmatrix}.$$

The block (i, j) for $2 \leq i \leq n$ and $1 \leq j \leq n$ of the displacement is then

$$(\nabla_{X_0, X_0}(U_1L_1 + U_2L_2))_{i,j} = -\mathcal{V}_i\mathcal{U}_{n+1-j} + \mathcal{U}_i\mathcal{V}_{n+1-j}.$$

The components of \mathcal{U}_i and \mathcal{V}_i are Toeplitz matrices (since they belong to \mathcal{R}), and therefore

$$\begin{aligned} (\nabla_{X_0, X_0}(U_1L_1 + U_2L_2))_{i,j} &= -\mathcal{V}_i\mathcal{U}_{n+1-j} + \mathcal{U}_i\mathcal{V}_{n+1-j} \\ &= \mathcal{U}_i\mathcal{V}_{n+1-j}^P - \mathcal{V}_i\mathcal{U}_{n+1-j}^P = (\mathcal{U}\mathcal{V}^P - \mathcal{V}\mathcal{U}^P)_{i,j}. \end{aligned}$$

The component $(1, j)$ for $1 \leq j < n$ is

$$(\nabla_{X_0, X_0}(U_1L_1 + U_2L_2))_{1,j} = \sum_{k=1}^{n-j} \mathcal{U}_{k+1}\mathcal{V}_{n+1-j-k} - \mathcal{V}_{k+1}\mathcal{U}_{n+1-j-k}.$$

Since the components of \mathcal{U} and \mathcal{V} are commutative, it follows that

$$\begin{aligned}
(\nabla_{X_0, X_0}(U_1 L_1 + U_2 L_2))_{1,j} &= \sum_{k=1}^{n-j} \mathcal{U}_{k+1} \mathcal{V}_{n+1-j-k} - \mathcal{V}_{k+1} \mathcal{U}_{n+1-j-k} \\
&= \sum_{k=1}^{n-j} \mathcal{U}_{k+1} \mathcal{V}_{n+1-j-k} - \mathcal{U}_{n+1-j-k} \mathcal{V}_{k+1} = \mathcal{U}_1 \mathcal{V}_{n+1-j} - \mathcal{V}_1 \mathcal{U}_{n+1-j} \\
&= (\mathcal{U} \mathcal{V}^P - \mathcal{V} \mathcal{U}^P)_{1,j}.
\end{aligned}$$

Finally, the component $(1, n)$ of the displacement is equal to $\mathbf{0}$. By the commutativity of the elements,

$$(\mathcal{U} \mathcal{V}^P - \mathcal{V} \mathcal{U}^P)_{1,n} = \mathcal{U}_1 \mathcal{V}_1 - \mathcal{V}_1 \mathcal{U}_1 = \mathbf{0}.$$

Therefore, $\nabla_{X_0, X_0}(B) = \nabla_{X_0, X_0}(U_1 L_1 + U_2 L_2)$. Since the nullspace of $\nabla_{X_0, X_0}(\cdot)$ is the space of all block-lower-triangular-Toeplitz matrices, B differs from $U_1 L_1 + U_2 L_2$ by at most an additive block-lower-triangular-Toeplitz matrix. However, the last block row of $U_1 L_1 + U_2 L_2$ satisfies:

$$\mathcal{E}_1^P(U_1 L_1 + U_2 L_2) = \mathcal{E}_1^P L_1 = \mathcal{U}^P.$$

By the persymmetry of B , it follows that $\mathcal{E}_1^P B = (B \mathcal{E}_1)^P$ and thus the first block column of B matches the first block column of $U_1 L_1 + U_2 L_2$. Since this is the case, $B = U_1 L_1 + U_2 L_2$. \square

Proof of Lemma 4.4. Since $GH^T = \sum_{j=1}^r G \mathcal{E}_j (H \mathcal{E}_j)^T$, it follows that

$$\begin{aligned}
\sum_{i=0}^{n-1} (X_0^T)^i GH^T X_0^i &= \sum_{i=0}^{n-1} (X_0^T)^i \left(\sum_{j=1}^r G \mathcal{E}_j (H \mathcal{E}_j)^T \right) X_0^i \\
&= \sum_{j=1}^r \sum_{i=0}^{n-1} (X_0^T)^i G \mathcal{E}_j (H \mathcal{E}_j)^T X_0^i,
\end{aligned}$$

and similarly

$$\sum_{i=0}^{n-1} X_0^i GH^T (X_0^T)^i = \sum_{j=1}^r \sum_{i=0}^{n-1} X_0^i G \mathcal{E}_j (H \mathcal{E}_j)^T (X_0^T)^i.$$

For any block column vectors $Q, R \in \mathbb{C}^{mn \times m}$ with blocks $Q_i, R_i \in \mathbb{C}^{m \times m}$,

$$\sum_{i=0}^{n-1} (X_0^T)^i QR^T X_0^i = \begin{bmatrix} Q_n & \cdots & Q_1 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & Q_n \end{bmatrix} \begin{bmatrix} R_n & \cdots & 0 \\ \vdots & \ddots & \vdots \\ R_1 & \cdots & R_n \end{bmatrix}.$$

Therefore, (74) holds. Similarly,

$$\sum_{i=0}^{n-1} X_0^i Q R^T (X_0^T)^i = \begin{bmatrix} Q_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ Q_n & \cdots & Q_1 \end{bmatrix} \begin{bmatrix} R_1 & \cdots & R_n \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R_1 \end{bmatrix}.$$

□

Proof of Corollary 4.2. From previous results,

$$\begin{aligned} \Delta_{X_0^T, X_0}(B) &= \begin{bmatrix} \dot{U} & \dot{V}_{[0]} \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & 0 \\ 0 & -\mathbf{I}_m \end{bmatrix} \begin{bmatrix} \mathcal{V}_{[0]}^P \\ \mathcal{U}^P \end{bmatrix} \\ &= \begin{bmatrix} \dot{U} & \dot{V}_{[0]} \end{bmatrix} A^{-1} A \begin{bmatrix} \mathbf{I}_m & 0 \\ 0 & -\mathbf{I}_m \end{bmatrix} A^P A^{-P} \begin{bmatrix} \mathcal{V}_{[0]}^P \\ \mathcal{U}^P \end{bmatrix} \\ &= G A \begin{bmatrix} \mathbf{I}_m & 0 \\ 0 & -\mathbf{I}_m \end{bmatrix} A^P H^T. \end{aligned}$$

The following identity then holds:

$$\begin{aligned} A \begin{bmatrix} \mathbf{I}_m & 0 \\ 0 & -\mathbf{I}_m \end{bmatrix} A^P &= \begin{bmatrix} A_{1,1} & -A_{1,2} \\ -A_{2,1} & A_{2,2} \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & 0 \\ 0 & -\mathbf{I}_m \end{bmatrix} \begin{bmatrix} A_{2,2}^P & -A_{1,2}^P \\ -A_{2,1}^P & A_{1,1}^P \end{bmatrix} \\ &= \begin{bmatrix} A_{1,1}A_{2,2}^P - A_{1,2}A_{2,1}^P & A_{1,1}A_{1,2}^P - A_{1,2}A_{1,1}^P \\ A_{2,1}A_{2,2}^P - A_{2,2}A_{2,1}^P & A_{2,2}A_{1,1}^P - A_{2,1}A_{1,2}^P \end{bmatrix} \\ &= \begin{bmatrix} A_{1,1}A_{2,2}^P - A_{1,2}A_{2,1}^P & \mathbf{0} \\ \mathbf{0} & A_{2,2}A_{1,1}^P - A_{2,1}A_{1,2}^P \end{bmatrix}. \end{aligned}$$

Therefore,

$$\Delta_{X_0^T, X_0}(B) = G \begin{bmatrix} A_{1,1}A_{2,2}^P - A_{1,2}A_{2,1}^P & \mathbf{0} \\ \mathbf{0} & -A_{2,2}A_{1,1}^P - A_{2,1}A_{1,2}^P \end{bmatrix} H^T.$$

The formula for B then follows from Theorem 4.2.

□

Proof of Theorem 4.3. Several relations can be proven easily with the Stein displacement results. Using Proposition 4.7,

$$\begin{aligned}
B\mathcal{E}_1 &= X_0 B X_0^T \mathcal{E}_1 + \dot{\mathcal{W}} \mathcal{X}^P \mathcal{E}_1 - \dot{\mathcal{X}} \mathcal{W}^P \mathcal{E}_1 \\
\mathcal{U} &= \dot{\mathcal{W}} \mathcal{X}_n^P - \dot{\mathcal{X}} \mathcal{W}_n^P = \dot{\mathcal{W}} \mathcal{X}_n^P - \dot{\mathcal{X}} \mathcal{U}_1 \\
B\mathcal{E}_n &= X_0^T B X_0 \mathcal{E}_n + \dot{\mathcal{U}} \mathcal{V}^P \mathcal{E}_n - \dot{\mathcal{V}} \mathcal{U}^P \mathcal{E}_n \\
\mathcal{W} &= \dot{\mathcal{U}} \mathcal{V}_1^P - \dot{\mathcal{V}} \mathcal{U}_1^P,
\end{aligned}$$

which proves (79) and (83). Similarly, using Proposition 4.5,

$$\begin{aligned}
X_0 B \mathcal{E}_n &= B X_0 \mathcal{E}_n + \mathcal{U} \mathcal{V}^P \mathcal{E}_n - \mathcal{V} \mathcal{U}^P \mathcal{E}_n \\
\dot{\mathcal{W}} &= \mathcal{U} \mathcal{V}_1^P - \mathcal{V} \mathcal{U}_1^P \\
X_0^T B \mathcal{E}_1 &= B X_0^T \mathcal{E}_1 + \mathcal{W} \mathcal{X}^P \mathcal{E}_1 - \mathcal{X} \mathcal{W}^P \mathcal{E}_1 \\
\dot{\mathcal{U}} &= \mathcal{W} \mathcal{X}_n^P - \mathcal{X} \mathcal{W}_n^P = \mathcal{W} \mathcal{X}_n^P - \mathcal{X} \mathcal{U}_1.
\end{aligned}$$

These equalities imply

$$\begin{aligned}
\mathcal{U} &= \mathcal{V} \mathcal{U}_1^P \mathcal{V}_1^{-P} + \dot{\mathcal{W}} \mathcal{V}_1^{-P}, \\
\mathcal{V} &= \mathcal{U} \mathcal{V}_1^P \mathcal{U}_1^{-P} - \dot{\mathcal{W}} \mathcal{U}_1^{-P}, \\
\mathcal{W} &= \dot{\mathcal{U}} \mathcal{X}_n^{-P} + \mathcal{X} \mathcal{U}_1 \mathcal{X}_n^{-P}, \quad \text{and} \\
\mathcal{X} &= -\dot{\mathcal{U}} \mathcal{U}_1^{-1} + \mathcal{W} \mathcal{X}_n^P \mathcal{U}_1^{-1},
\end{aligned}$$

which prove (77), (80), (84), and (87).

Equating (77) and (79), which have each been proven true, yields

$$\begin{aligned}
\dot{\mathcal{W}} \mathcal{X}_n^P - \dot{\mathcal{X}} \mathcal{U}_1 &= \mathcal{V} \mathcal{U}_1^P \mathcal{V}_1^{-P} + \dot{\mathcal{W}} \mathcal{V}_1^{-P} \\
\mathcal{V} \mathcal{U}_1^P \mathcal{V}_1^{-P} &= \dot{\mathcal{W}} (\mathcal{X}_n^P - \mathcal{V}_1^{-P}) - \dot{\mathcal{X}} \mathcal{U}_1 \\
\mathcal{V} \mathcal{U}_1^P \mathcal{V}_1^{-P} &= \dot{\mathcal{W}} (\mathcal{X}_n^P \mathcal{V}_1^P - \mathbf{I}_m) \mathcal{V}_1^{-P} - \dot{\mathcal{X}} \mathcal{U}_1 \\
\mathcal{V} &= \dot{\mathcal{W}} (\mathcal{X}_n^P \mathcal{V}_1^P - \mathbf{I}_m) \mathcal{U}_1^{-P} - \dot{\mathcal{X}} \mathcal{U}_1 \mathcal{V}_1^P \mathcal{U}_1^{-P}.
\end{aligned}$$

It is evident from expanding the displacement that for any (m, n) two-level matrix Q , $[\nabla_{X_0, X_0}(Q)]_{1, n} = \mathbf{0}$ (see the proof of Lemma 4.3). Since $\nabla_{X_0, X_0}(B) = \mathcal{U}\mathcal{V}^P - \mathcal{V}\mathcal{U}^P$, it follows that $\mathcal{U}_1\mathcal{V}_1^P = \mathcal{V}_1\mathcal{U}_1^P$. Therefore:

$$\begin{aligned}\mathcal{V} &= \dot{\mathcal{W}}(\mathcal{X}_n^P\mathcal{V}_1^{-P} - \mathbf{I}_m)\mathcal{U}_1^{-P} - \dot{\mathcal{X}}\mathcal{V}_1\mathcal{U}_1^P\mathcal{U}_1^{-P} \\ &= -\dot{\mathcal{W}}\Lambda^{-P} - \dot{\mathcal{X}}\mathcal{V}_1,\end{aligned}$$

which proves (82).

Similarly, equating (83) and (84) yields

$$\begin{aligned}\dot{\mathcal{U}}\mathcal{V}_1^P - \dot{\mathcal{V}}\mathcal{U}_1^P &= \dot{\mathcal{U}}\mathcal{X}_n^{-P} + \mathcal{X}\mathcal{U}_1\mathcal{X}_n^{-P} \\ \mathcal{X}\mathcal{U}_1\mathcal{X}_n^{-P} &= \dot{\mathcal{U}}(\mathcal{V}_1^P - \mathcal{X}_n^{-P}) - \dot{\mathcal{V}}\mathcal{U}_1^P \\ \mathcal{X}\mathcal{U}_1\mathcal{X}_n^{-P} &= \dot{\mathcal{U}}(\mathcal{V}_1^P\mathcal{X}_n^P - \mathbf{I}_m)\mathcal{X}_n^{-P} - \dot{\mathcal{V}}\mathcal{U}_1^P \\ \mathcal{X} &= \dot{\mathcal{U}}(\mathcal{V}_1^P\mathcal{X}_n^P - \mathbf{I}_m)\mathcal{U}_1^{-1} - \dot{\mathcal{V}}\mathcal{U}_1^P\mathcal{X}_n^P\mathcal{U}_1^{-1}.\end{aligned}$$

It is evident from expanding the displacement that for any (m, n) two-level matrix Q , $[\nabla_{X_0^T, X_0^T}(Q)]_{n, 1} = \mathbf{0}$ (see the proof of Lemma 4.3). Since $\nabla_{X_0^T, X_0^T}(B) = \mathcal{W}\mathcal{X}^P - \mathcal{W}\mathcal{X}^P$, it follows that $\mathcal{W}_n\mathcal{X}_n^P = \mathcal{X}_n\mathcal{W}_n^P$. Taking into account that $\mathcal{W}_n^P = \mathcal{U}_1$, it follows that

$$\begin{aligned}\mathcal{X} &= \dot{\mathcal{U}}(\mathcal{V}_1^P\mathcal{X}_n^P - \mathbf{I}_m)\mathcal{U}_1^{-1} - \dot{\mathcal{V}}\mathcal{X}_n\mathcal{U}_1\mathcal{U}_1^{-1} \\ &= \dot{\mathcal{U}}(\mathcal{V}_1^P\mathcal{X}_n^P - \mathbf{I}_m)\mathcal{U}_1^{-1} - \dot{\mathcal{V}}\mathcal{X}_n.\end{aligned}$$

Next, the following identities hold:

$$\begin{aligned}\mathcal{V}_1^P\mathcal{X}_n^P\mathcal{U}_1^{-1} &= \mathcal{V}_1^P\mathcal{W}_n^{-1}\mathcal{W}_n\mathcal{X}_n^P\mathcal{U}_1^{-1} = \mathcal{V}_1^P\mathcal{W}_n^{-1}\mathcal{X}_n\mathcal{W}_n^P\mathcal{U}_1^{-1} \\ &= \mathcal{V}_1^P\mathcal{U}_1^{-P}\mathcal{X}_n = \mathcal{U}_1^{-1}\mathcal{U}_1\mathcal{V}_1^P\mathcal{U}_1^{-P}\mathcal{X}_n \\ &= \mathcal{U}_1^{-1}\mathcal{V}_1\mathcal{U}_1^P\mathcal{U}_1^{-P}\mathcal{X}_n = \mathcal{U}_1^{-1}\mathcal{V}_1\mathcal{X}_n.\end{aligned}$$

This result implies

$$\begin{aligned}\mathcal{X} &= \dot{\mathcal{U}}\mathcal{U}_1^{-1}(\mathcal{V}_1\mathcal{X}_n - \mathbf{I}_m) - \dot{\mathcal{V}}\mathcal{X}_n \\ &= -\dot{\mathcal{U}}\Lambda^{-1} - \dot{\mathcal{V}}\mathcal{X}_n,\end{aligned}$$

which proves (86).

After establishing the equality

$$\begin{aligned}\mathcal{U}_1 \mathcal{V}_1^P &= \mathcal{V}_1 \mathcal{U}_1^P \\ \mathcal{U}_1 &= \mathcal{V}_1 \mathcal{U}_1^P \mathcal{V}_1^{-P} \\ \mathcal{V}_1^{-1} \mathcal{U}_1 &= \mathcal{U}_1^P \mathcal{V}_1^{-P},\end{aligned}$$

it is possible to prove the remaining identities of (78), (81), (85), and (88):

$$\begin{aligned}\mathcal{U} &= \dot{\mathcal{W}} \mathcal{X}_n^P - \dot{\mathcal{X}} \mathcal{U}_1 = (\mathcal{U} \mathcal{V}_1^P - \mathcal{V}_1 \mathcal{U}_1^P) \mathcal{X}_n^P - \dot{\mathcal{X}} \mathcal{U}_1 \\ &= -\mathcal{V}_1 \mathcal{U}_1^P \mathcal{X}_n^P (\mathbf{I}_m - \mathcal{V}_1^P \mathcal{X}_n^P)^{-1} - \dot{\mathcal{X}} \mathcal{U}_1 (\mathbf{I}_m - \mathcal{V}_1^P \mathcal{X}_n^P)^{-1} \\ &= -\mathcal{V} (\mathcal{X}_n^{-P} \mathcal{U}_1^{-P} - \mathcal{V}_1^P \mathcal{X}_n^P \mathcal{X}_n^{-P} \mathcal{U}_1^{-P})^{-1} - \dot{\mathcal{X}} (\mathcal{U}_1^{-1} - \mathcal{V}_1^P \mathcal{X}_n^P \mathcal{U}_1^{-1})^{-1} \\ &= -\mathcal{V} (\mathcal{U}_1^{-1} \mathcal{X}_n^{-1} - \mathcal{U}_1^{-1} \mathcal{V}_1)^{-1} - \dot{\mathcal{X}} (\mathcal{U}_1^{-1} - \mathcal{U}_1^{-1} \mathcal{V}_1 \mathcal{X}_n)^{-1} \\ &= -\mathcal{V} \mathcal{X}_n (\mathbf{I}_m - \mathcal{V}_1 \mathcal{X}_n)^{-1} \mathcal{U}_1 - \dot{\mathcal{X}} (\mathbf{I}_m - \mathcal{V}_1 \mathcal{X}_n)^{-1} \mathcal{U}_1 \\ &= -\mathcal{V} \mathcal{X}_n \Lambda - \dot{\mathcal{X}} \Lambda\end{aligned}$$

$$\begin{aligned}\mathcal{V} &= -\dot{\mathcal{W}} \Lambda^{-P} - \dot{\mathcal{X}} \mathcal{V}_1 = -(\mathcal{U} \mathcal{X}_n^{-P} + \dot{\mathcal{X}} \mathcal{U}_1 \mathcal{X}_n^{-P}) \Lambda^{-P} - \dot{\mathcal{X}} \mathcal{V}_1 \\ &= -\mathcal{U} \mathcal{X}_n^{-P} \Lambda^{-P} - \dot{\mathcal{X}} (\mathcal{U}_1 \mathcal{X}_n^{-P} \Lambda^{-P} + \mathcal{V}_1) \\ &= -\mathcal{U} (\mathcal{X}_n \Lambda)^{-P} - \dot{\mathcal{X}} (\mathcal{U}_1 \mathcal{X}_n^{-P} (\mathbf{I}_m - \mathcal{X}_n^P \mathcal{V}_1^P) \mathcal{U}_1^{-P} + \mathcal{V}_1) \\ &= -\mathcal{U} (\mathcal{X}_n \Lambda)^{-P} - \dot{\mathcal{X}} (\mathcal{U}_1 (\mathcal{X}_n^{-P} - \mathcal{V}_1^P) \mathcal{U}_1^{-P} + \mathcal{V}_1) \\ &= -\mathcal{U} (\mathcal{X}_n \Lambda)^{-P} - \dot{\mathcal{X}} (\mathcal{U}_1 \mathcal{X}_n^{-P} \mathcal{U}_1^{-P} - \mathcal{V}_1 \mathcal{U}_1^P \mathcal{U}_1^{-P} + \mathcal{V}_1) \\ &= -\mathcal{U} (\mathcal{X}_n \Lambda)^{-P} - \dot{\mathcal{X}} (\mathcal{U}_1 \mathcal{U}_1^{-1} \mathcal{X}_n^{-1}) = -\mathcal{U} (\mathcal{X}_n \Lambda)^{-P} - \dot{\mathcal{X}} \mathcal{X}_n^{-1}\end{aligned}$$

$$\begin{aligned}\mathcal{W} &= \dot{\mathcal{U}} \mathcal{V}_1^P - \dot{\mathcal{V}} \mathcal{U}_1^P = -(\dot{\mathcal{V}} \mathcal{X}_n + \mathcal{X}) \Lambda \mathcal{V}_1^P - \dot{\mathcal{V}} \mathcal{U}_1^P \\ &= -\dot{\mathcal{V}} (\mathcal{X}_n \Lambda \mathcal{V}_1^P + \mathcal{U}_1^P) - \mathcal{X} \Lambda \mathcal{V}_1^P \\ &= -\dot{\mathcal{V}} (\mathcal{X}_n (\mathbf{I}_m - \mathcal{V}_1 \mathcal{X}_n)^{-1} \mathcal{U}_1 \mathcal{V}_1^P + \mathcal{U}_1^P) - \mathcal{X} \Lambda \mathcal{V}_1^P \\ &= -\dot{\mathcal{V}} (\mathcal{X}_n (\mathbf{I}_m - \mathcal{V}_1 \mathcal{X}_n)^{-1} \mathcal{V}_1 + \mathbf{I}_m) \mathcal{U}_1^P - \mathcal{X} \Lambda \mathcal{V}_1^P \\ &= -\dot{\mathcal{V}} ((\mathcal{X}_n^{-1} - \mathcal{V}_1)^{-1} \mathcal{V}_1 + \mathbf{I}_m) \mathcal{U}_1^P - \mathcal{X} \Lambda \mathcal{V}_1^P \\ &= -\dot{\mathcal{V}} (\mathbf{I}_m - \mathcal{X}_n \mathcal{V}_1)^{-1} \mathcal{X}_n \mathcal{V}_1 + \mathbf{I}_m) \mathcal{U}_1^P - \mathcal{X} \Lambda \mathcal{V}_1^P\end{aligned}$$

$$\begin{aligned}
&= -\dot{\mathcal{V}}(\mathbf{I}_m - \mathcal{X}_n \mathcal{V}_1)^{-1} (\mathcal{X}_n \mathcal{V}_1 + \mathbf{I}_m - \mathcal{X}_n \mathcal{V}_1) \mathcal{U}_1^P - \mathcal{X} \Lambda \mathcal{V}_1^P \\
&= -\dot{\mathcal{V}}(\mathbf{I}_m - \mathcal{X}_n \mathcal{V}_1)^{-1} \mathcal{U}_1^P - \mathcal{X} \Lambda \mathcal{V}_1^P \\
&= -\dot{\mathcal{V}}(\mathcal{U}_1^{-P} - \mathcal{U}_1^{-P} \mathcal{X}_n \mathcal{V}_1)^{-1} - \mathcal{X} \Lambda \mathcal{V}_1^P \\
&= -\dot{\mathcal{V}}(\mathcal{U}_1^{-P} - \mathcal{X}_n^P \mathcal{U}_1^{-1} \mathcal{V}_1)^{-1} - \mathcal{X} \Lambda \mathcal{V}_1^P \\
&= -\dot{\mathcal{V}}(\mathcal{U}_1^{-P} - \mathcal{X}_n^P \mathcal{V}_1^P \mathcal{U}_1^{-P})^{-1} - \mathcal{X} \Lambda \mathcal{V}_1^P \\
&= -\dot{\mathcal{V}} \mathcal{U}_1^P (\mathbf{I}_m - \mathcal{X}_n^P \mathcal{V}_1^P)^{-1} - \mathcal{X} \Lambda \mathcal{V}_1^P = -\dot{\mathcal{V}} \Lambda^P - \mathcal{X} \Lambda \mathcal{V}_1^P \\
\mathcal{X} &= -\dot{\mathcal{U}} \Lambda^{-1} - \dot{\mathcal{V}} \mathcal{X}_n = -(\mathcal{W} \mathcal{V}_1^{-P} + \dot{\mathcal{V}} \mathcal{U}_1^P \mathcal{V}_1^{-P}) \Lambda^{-1} - \dot{\mathcal{V}} \mathcal{X}_n \\
&= -\dot{\mathcal{V}}(\mathcal{U}_1^P \mathcal{V}_1^{-P} \Lambda^{-1} + \mathcal{X}_n) - \mathcal{W} \mathcal{V}_1^{-P} \Lambda^{-1} \\
&= -\dot{\mathcal{V}}(\mathcal{U}_1^P \mathcal{V}_1^{-P} \mathcal{U}_1^{-1} (\mathbf{I}_m - \mathcal{V}_1 \mathcal{X}_n) + \mathcal{X}_n) - \mathcal{W} \mathcal{V}_1^{-P} \Lambda^{-1} \\
&= -\dot{\mathcal{V}}(\mathcal{V}_1^{-1} \mathcal{U}_1 \mathcal{U}_1^{-1} (\mathbf{I}_m - \mathcal{V}_1 \mathcal{X}_n) + \mathcal{X}_n) - \mathcal{W} \mathcal{V}_1^{-P} \Lambda^{-1} \\
&= -\dot{\mathcal{V}}(\mathcal{V}_1^{-1} - \mathcal{X}_n + \mathcal{X}_n) - \mathcal{W} \mathcal{V}_1^{-P} \Lambda^{-1} = -\dot{\mathcal{V}} \mathcal{V}_1^{-1} - \mathcal{W} \mathcal{V}_1^{-P} \Lambda^{-1}.
\end{aligned}$$

These equalities complete the proof. \square

Proof of Theorem 4.4. From Corollary 4.1, it suffices for each point to show that the Stein displacement of B is equal to the term inside $\mathfrak{S}_B(\cdot)$.

1. This formula is a block-level version of the Barnett formula, and it follows from Proposition 4.7; that is,

$$\Delta_{X_0^T, X_0}(B) = \begin{bmatrix} \dot{\mathcal{U}} & \dot{\mathcal{V}} \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} \begin{bmatrix} \mathcal{V}^P \\ \mathcal{U}^P \end{bmatrix},$$

and therefore by Corollary 4.1, $B = \mathfrak{S}_B(GDH^T)$.

2. This formula is the original Gohberg-Heinig formula. From Theorem 4.3, the following identities hold:

$$\begin{bmatrix} \dot{\mathcal{U}} & \dot{\mathcal{V}} \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & \mathcal{V}_1^P \\ \mathbf{0} & -\mathcal{U}_1^P \end{bmatrix} = \begin{bmatrix} \dot{\mathcal{U}} & \mathcal{W} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -\mathcal{U}_1 & \mathcal{V}_1 \\ \mathbf{0} & \mathbf{I}_m \end{bmatrix} \begin{bmatrix} \mathcal{V}^P \\ \mathcal{U}^P \end{bmatrix} = \begin{bmatrix} \dot{\mathcal{W}}^P \\ \mathcal{U}^P \end{bmatrix}.$$

Let $A^{-1} = \begin{bmatrix} \mathbf{I}_m & \mathcal{V}_1^P \\ \mathbf{0} & -\mathcal{U}_1^P \end{bmatrix}$; then using a blockwise matrix inversion, $A = \begin{bmatrix} \mathbf{I}_m & \mathcal{V}_1^P \mathcal{U}_1^{-P} \\ \mathbf{0} & -\mathcal{U}_1^{-P} \end{bmatrix}$.

Using this definition, the following identity holds:

$$\begin{aligned} A \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} A^P &= \begin{bmatrix} \mathbf{I}_m & \mathcal{V}_1^P \mathcal{U}_1^{-P} \\ \mathbf{0} & -\mathcal{U}_1^{-P} \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} \begin{bmatrix} -\mathcal{U}_1^{-1} & \mathcal{U}_1^{-1} \mathcal{V}_1 \\ \mathbf{0} & \mathbf{I}_m \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I}_m & -\mathcal{V}_1^P \mathcal{U}_1^{-P} \\ \mathbf{0} & \mathcal{U}_1^{-P} \end{bmatrix} \begin{bmatrix} -\mathcal{U}_1^{-1} & \mathcal{U}_1^{-1} \mathcal{V}_1 \\ \mathbf{0} & \mathbf{I}_m \end{bmatrix} \\ &= \begin{bmatrix} -\mathcal{U}_1^{-1} & \mathcal{U}_1^{-1} \mathcal{V}_1 - \mathcal{V}_1^P \mathcal{U}_1^{-P} \\ \mathbf{0} & -\mathcal{U}_1^{-P} \end{bmatrix} = \begin{bmatrix} -\mathcal{U}_1^{-1} & \mathbf{0} \\ \mathbf{0} & -\mathcal{U}_1^{-P} \end{bmatrix} = D. \end{aligned}$$

From the proof of Corollary 4.2, then,

$$\begin{aligned} \Delta_{x_0^T, x_0}(B) &= \begin{bmatrix} \dot{\mathcal{U}} & \dot{\mathcal{V}} \end{bmatrix} A^{-1} A \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} A^P A^{-P} \begin{bmatrix} \mathcal{V}^P \\ \mathcal{U}^P \end{bmatrix} \\ &= \begin{bmatrix} \dot{\mathcal{U}} & \mathcal{W} \end{bmatrix} \begin{bmatrix} -\mathcal{U}_1^{-1} & \mathbf{0} \\ \mathbf{0} & -\mathcal{U}_1^{-P} \end{bmatrix} \begin{bmatrix} \dot{\mathcal{W}}^P \\ \mathcal{U}^P \end{bmatrix} = GDH^T, \end{aligned}$$

and therefore by Corollary 4.1, $B = \mathfrak{G}_B(GDH^T)$.

3. From Theorem 4.3, the following identities hold:

$$\begin{bmatrix} \dot{\mathcal{U}} & \dot{\mathcal{V}} \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & -\Lambda^{-1} \\ \mathbf{0} & -\mathcal{X}_n \end{bmatrix} = \begin{bmatrix} \dot{\mathcal{U}} & \mathcal{X} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -\mathcal{X}_n^P & -\Lambda^{-P} \\ \mathbf{0} & \mathbf{I}_m \end{bmatrix} \begin{bmatrix} \mathcal{V}^P \\ \mathcal{U}^P \end{bmatrix} = \begin{bmatrix} \dot{\mathcal{X}}^P \\ \mathcal{U}^P \end{bmatrix}.$$

Let $A^{-1} = \begin{bmatrix} \mathbf{I}_m & -\Lambda^{-1} \\ \mathbf{0} & -\mathcal{X}_n \end{bmatrix}$; then using a blockwise matrix inversion, $A = \begin{bmatrix} \mathbf{I}_m & -\Lambda^{-1} \mathcal{X}_n^{-1} \\ \mathbf{0} & -\mathcal{X}_n^{-1} \end{bmatrix}$.

Using this definition, the following identity holds:

$$\begin{aligned}
A \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} A^P &= \begin{bmatrix} \mathbf{I}_m & -\Lambda^{-1} \mathcal{X}_n^{-1} \\ \mathbf{0} & -\mathcal{X}_n^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} \begin{bmatrix} -\mathcal{X}_n^{-P} & -\mathcal{X}_n^{-P} \Lambda^{-P} \\ \mathbf{0} & \mathbf{I}_m \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{I}_m & \Lambda^{-1} \mathcal{X}_n^{-1} \\ \mathbf{0} & \mathcal{X}_n^{-1} \end{bmatrix} \begin{bmatrix} -\mathcal{X}_n^{-P} & -\mathcal{X}_n^{-P} \Lambda^{-P} \\ \mathbf{0} & \mathbf{I}_m \end{bmatrix} \\
&= \begin{bmatrix} -\mathcal{X}_n^{-P} & \Lambda^{-1} \mathcal{X}_n^{-1} - \mathcal{X}_n^{-P} \Lambda^{-P} \\ \mathbf{0} & \mathcal{X}_n^{-1} \end{bmatrix}.
\end{aligned}$$

From the proof of Theorem 4.3,

$$\begin{aligned}
\Lambda^{-1} \mathcal{X}_n^{-1} &= \mathcal{U}_1^{-1} (\mathbf{I}_m - \mathcal{V}_1 \mathcal{X}_n) \mathcal{X}_n^{-1} = \mathcal{U}_1^{-1} \mathcal{X}_n^{-1} (\mathbf{I}_m - \mathcal{X}_n \mathcal{V}_1) \\
&= \mathcal{X}_n^{-P} \mathcal{U}_1^{-P} (\mathbf{I}_m - \mathcal{X}_n \mathcal{W}_n^P \mathcal{W}_n^{-P} \mathcal{V}_1) = \mathcal{X}_n^{-P} \mathcal{U}_1^{-P} (\mathbf{I}_m - \mathcal{W}_n \mathcal{X}_n^P \mathcal{U}_1^{-1} \mathcal{V}_1) \\
&= \mathcal{X}_n^{-P} (\mathcal{U}_1^{-P} - \mathcal{W}_n \mathcal{X}_n^P \mathcal{V}_1^P \mathcal{U}_1^{-P}) = \mathcal{X}_n^{-P} (\mathbf{I}_m - \mathcal{W}_n \mathcal{X}_n^P \mathcal{V}_1^P) \mathcal{U}_1^{-P} = \mathcal{X}_n^{-P} \Lambda^{-P}.
\end{aligned}$$

Therefore,

$$A \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} A^P = \begin{bmatrix} -\mathcal{X}_n^{-P} & \mathbf{0} \\ \mathbf{0} & \mathcal{X}_n^{-1} \end{bmatrix} = D.$$

From the proof of Corollary 4.2, then,

$$\begin{aligned}
\Delta_{X_0^r, X_0}(B) &= \begin{bmatrix} \dot{\mathcal{U}} & \dot{\mathcal{V}} \end{bmatrix} A^{-1} A \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} A^P A^{-P} \begin{bmatrix} \mathcal{V}^P \\ \mathcal{U}^P \end{bmatrix} \\
&= \begin{bmatrix} \dot{\mathcal{U}} & \mathcal{X} \end{bmatrix} \begin{bmatrix} -\mathcal{X}_n^{-P} & \mathbf{0} \\ \mathbf{0} & \mathcal{X}_n^{-1} \end{bmatrix} \begin{bmatrix} \dot{\mathcal{X}}^P \\ \mathcal{U}^P \end{bmatrix} = GDH^T,
\end{aligned}$$

and thus by Corollary 4.1 it follows that $B = \mathfrak{G}_B(GDH^T)$.

4. From Theorem 4.3, the following identities hold:

$$\begin{bmatrix} \dot{\mathcal{U}} & \dot{\mathcal{V}} \end{bmatrix} \begin{bmatrix} \mathcal{V}_1^P & \mathbf{0} \\ -\mathcal{U}_1^P & \mathbf{I}_m \end{bmatrix} = \begin{bmatrix} \mathcal{W} & \dot{\mathcal{V}} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ -\mathcal{U}_1 & \mathcal{V}_1 \end{bmatrix} \begin{bmatrix} \mathcal{V}^P \\ \mathcal{U}^P \end{bmatrix} = \begin{bmatrix} \mathcal{V}^P \\ \dot{\mathcal{W}}^P \end{bmatrix}.$$

Let $A^{-1} = \begin{bmatrix} \mathcal{V}_1^P & \mathbf{0} \\ -\mathcal{U}_1^P & \mathbf{I}_m \end{bmatrix}$; then using a blockwise matrix inversion, $A = \begin{bmatrix} \mathcal{V}_1^{-P} & \mathbf{0} \\ \mathcal{U}_1^P \mathcal{V}_1^{-P} & \mathbf{I}_m \end{bmatrix}$.

Using this definition,

$$\begin{aligned} A \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} A^P &= \begin{bmatrix} \mathcal{V}_1^{-P} & \mathbf{0} \\ \mathcal{U}_1^P \mathcal{V}_1^{-P} & \mathbf{I}_m \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathcal{V}_1^{-1} \mathcal{U}_1 & \mathcal{V}_1^{-1} \end{bmatrix} \\ &= \begin{bmatrix} \mathcal{V}_1^{-P} & \mathbf{0} \\ \mathcal{U}_1^P \mathcal{V}_1^{-P} & -\mathbf{I}_m \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathcal{V}_1^{-1} \mathcal{U}_1 & \mathcal{V}_1^{-1} \end{bmatrix} \\ &= \begin{bmatrix} \mathcal{V}_1^{-P} & \mathbf{0} \\ \mathcal{U}_1^P \mathcal{V}_1^{-P} - \mathcal{V}_1^{-1} \mathcal{U}_1 & -\mathcal{V}_1^{-1} \end{bmatrix} = \begin{bmatrix} \mathcal{V}_1^{-P} & \mathbf{0} \\ \mathbf{0} & -\mathcal{V}_1^{-1} \end{bmatrix} = D. \end{aligned}$$

From the proof of Corollary 4.2, then,

$$\begin{aligned} \Delta_{X_0^T, X_0}(B) &= \begin{bmatrix} \dot{\mathcal{U}} & \dot{\mathcal{V}} \end{bmatrix} A^{-1} A \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} A^P A^{-P} \begin{bmatrix} \mathcal{V}^P \\ \mathcal{U}^P \end{bmatrix} \\ &= \begin{bmatrix} \mathcal{W} & \dot{\mathcal{V}} \end{bmatrix} \begin{bmatrix} \mathcal{V}_1^{-P} & \mathbf{0} \\ \mathbf{0} & -\mathcal{V}_1^{-1} \end{bmatrix} \begin{bmatrix} \mathcal{V}^P \\ \dot{\mathcal{W}}^P \end{bmatrix} = GDH^T, \end{aligned}$$

and therefore by Corollary 4.1, $B = \mathfrak{G}_B(GDH^T)$.

5. From Theorem 4.3, the following identities hold:

$$\begin{bmatrix} \dot{\mathcal{U}} & \dot{\mathcal{V}} \end{bmatrix} \begin{bmatrix} -\Lambda^{-1} & \mathbf{0} \\ -\mathcal{X}_n & \mathbf{I}_m \end{bmatrix} = \begin{bmatrix} \mathcal{X} & \dot{\mathcal{V}} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ -\mathcal{X}_n^P & -\Lambda^{-P} \end{bmatrix} \begin{bmatrix} \mathcal{V}^P \\ \mathcal{U}^P \end{bmatrix} = \begin{bmatrix} \mathcal{V}^P \\ \dot{\mathcal{X}}^P \end{bmatrix}.$$

Let $A^{-1} = \begin{bmatrix} -\Lambda^{-1} & \mathbf{0} \\ -\mathcal{X}_n & \mathbf{I}_m \end{bmatrix}$; then using a blockwise matrix inversion, $A = \begin{bmatrix} -\Lambda & \mathbf{0} \\ -\mathcal{X}_n \Lambda & \mathbf{I}_m \end{bmatrix}$.

Using this definition,

$$\begin{aligned} A \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} A^P &= \begin{bmatrix} -\Lambda & \mathbf{0} \\ -\mathcal{X}_n \Lambda & \mathbf{I}_m \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ -\Lambda^P \mathcal{X}_n^P & -\Lambda^P \end{bmatrix} \\ &= \begin{bmatrix} -\Lambda & \mathbf{0} \\ -\mathcal{X}_n \Lambda & -\mathbf{I}_m \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ -\Lambda^P \mathcal{X}_n^P & -\Lambda^P \end{bmatrix} = \begin{bmatrix} -\Lambda & \mathbf{0} \\ \Lambda^P \mathcal{X}_n^P - \mathcal{X}_n \Lambda & \Lambda^P \end{bmatrix}. \end{aligned}$$

From the proof of point 3, since $\Lambda^{-1}\mathcal{X}_n^{-1} = \mathcal{X}_n^{-P}\Lambda^{-P}$, it must also be true that $\Lambda^P\mathcal{X}_n^P = \mathcal{X}_n\Lambda$, and therefore

$$A \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} A^P = \begin{bmatrix} -\Lambda & \mathbf{0} \\ \mathbf{0} & \Lambda^P \end{bmatrix} = D.$$

From the proof of Corollary 4.2, then,

$$\begin{aligned} \Delta_{\mathcal{X}_0^T, \mathcal{X}_0}(B) &= \begin{bmatrix} \dot{\mathcal{U}} & \dot{\mathcal{V}} \end{bmatrix} A^{-1} A \begin{bmatrix} \mathbf{I}_m & 0 \\ 0 & -\mathbf{I}_m \end{bmatrix} A^P A^{-P} \begin{bmatrix} \mathcal{V}^P \\ \mathcal{U}^P \end{bmatrix} \\ &= \begin{bmatrix} \mathcal{X} & \dot{\mathcal{V}} \end{bmatrix} \begin{bmatrix} -\Lambda & \mathbf{0} \\ \mathbf{0} & \Lambda^P \end{bmatrix} \begin{bmatrix} \mathcal{V}^P \\ \dot{\mathcal{X}}^P \end{bmatrix} = GDH^T, \end{aligned}$$

and therefore by Corollary 4.1, $B = \mathfrak{G}_B(GDH^T)$.

6. From Theorem 4.3, the following identities hold:

$$\begin{bmatrix} \dot{\mathcal{U}} & \dot{\mathcal{V}} \end{bmatrix} \begin{bmatrix} \mathcal{V}_1^P & -\Lambda^{-1} \\ -\mathcal{U}_1^P & -\mathcal{X}_n \end{bmatrix} = \begin{bmatrix} \mathcal{W} & \mathcal{X} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -\mathcal{X}_n^P & -\Lambda^{-P} \\ -\mathcal{U}_1 & \mathcal{V}_1 \end{bmatrix} \begin{bmatrix} \mathcal{V}^P \\ \mathcal{U}^P \end{bmatrix} = \begin{bmatrix} \dot{\mathcal{X}}^P \\ \dot{\mathcal{W}}^P \end{bmatrix}.$$

Let $A^{-1} = \begin{bmatrix} \mathcal{V}_1^P & -\Lambda^{-1} \\ -\mathcal{U}_1^P & -\mathcal{X}_n \end{bmatrix}$. Let $\Theta = (\mathcal{V}_1^P + \Lambda^{-1}\mathcal{X}_n^{-1}\mathcal{U}_1^P)$; then using a blockwise inversion,

$$A = \begin{bmatrix} \Theta^{-1} & -\Theta^{-1}\Lambda^{-1}\mathcal{X}_n^{-1} \\ -\mathcal{X}_n^{-1}\mathcal{U}_1^P\Theta^{-1} & -\mathcal{X}_n^{-1} + \mathcal{X}_n^{-1}\mathcal{U}_1^P\Theta^{-1}\Lambda^{-1}\mathcal{X}_n^{-1} \end{bmatrix}.$$

Manipulations show

$$\begin{aligned} \mathcal{V}_1^P + \Lambda^{-1}\mathcal{X}_n^{-1}\mathcal{U}_1^P &= \mathcal{V}_1^P + \mathcal{U}_1^{-1}(\mathbf{I}_m - \mathcal{V}_1\mathcal{X}_n)\mathcal{X}_n^{-1}\mathcal{U}_1^P \\ &= \mathcal{V}_1^P + \mathcal{U}_1^{-1}\mathcal{X}_n^{-1}\mathcal{U}_1^P - \mathcal{U}_1^{-1}\mathcal{V}_1\mathcal{U}_1^P \\ &= \mathcal{V}_1^P + \mathcal{X}_n^{-P}\mathcal{U}_1^{-P}\mathcal{U}_1^P - \mathcal{U}_1^{-1}\mathcal{U}_1\mathcal{V}_1^P \\ &= \mathcal{V}_1^P + \mathcal{X}_n^{-P} - \mathcal{V}_1^P = \mathcal{X}_n^{-P}, \end{aligned}$$

and therefore

$$\begin{aligned}
A &= \begin{bmatrix} \mathcal{X}_n^P & -\mathcal{X}_n^P \Lambda^{-1} \mathcal{X}_n^{-1} \\ -\mathcal{X}_n^{-1} \mathcal{U}_1^P \mathcal{X}_n^P & -\mathcal{X}_n^{-1} + \mathcal{X}_n^{-1} \mathcal{U}_1^P \mathcal{X}_n^P \Lambda^{-1} \mathcal{X}_n^{-1} \end{bmatrix} \\
&= \begin{bmatrix} \mathcal{X}_n^P & -\Lambda^{-P} \mathcal{X}_n \mathcal{X}_n^{-1} \\ -\mathcal{X}_n^{-1} \mathcal{X}_n \mathcal{U}_1 & -\mathcal{X}_n^{-1} + \mathcal{X}_n^{-1} \mathcal{X}_n \mathcal{U}_1 \Lambda^{-1} \mathcal{X}_n^{-1} \end{bmatrix} \\
&= \begin{bmatrix} \mathcal{X}_n^P & -\Lambda^{-P} \\ -\mathcal{U}_1 & -\mathcal{X}_n^{-1} + \mathcal{U}_1 \Lambda^{-1} \mathcal{X}_n^{-1} \end{bmatrix} = \begin{bmatrix} \mathcal{X}_n^P & -\Lambda^{-P} \\ -\mathcal{U}_1 & -(\mathbf{I}_m - \mathcal{U}_1 \Lambda^{-1}) \mathcal{X}_n^{-1} \end{bmatrix} \\
&= \begin{bmatrix} \mathcal{X}_n^P & -\Lambda^{-P} \\ -\mathcal{U}_1 & -(\mathbf{I}_m - \mathbf{I}_m + \mathcal{V}_1 \mathcal{X}_n) \mathcal{X}_n^{-1} \end{bmatrix} = \begin{bmatrix} \mathcal{X}_n^P & -\Lambda^{-P} \\ -\mathcal{U}_1 & -\mathcal{V}_1 \end{bmatrix}.
\end{aligned}$$

Using this definition,

$$\begin{aligned}
A \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} A^P &= \begin{bmatrix} \mathcal{X}_n^P & -\Lambda^{-P} \\ -\mathcal{U}_1 & -\mathcal{V}_1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} \begin{bmatrix} -\mathcal{V}_1^P & -\Lambda^{-1} \\ -\mathcal{U}_1^P & \mathcal{X}_n \end{bmatrix} \\
&= \begin{bmatrix} \mathcal{X}_n^P & \Lambda^{-P} \\ -\mathcal{U}_1 & \mathcal{V}_1 \end{bmatrix} \begin{bmatrix} -\mathcal{V}_1^P & -\Lambda^{-1} \\ -\mathcal{U}_1^P & \mathcal{X}_n \end{bmatrix} \\
&= \begin{bmatrix} -\mathcal{X}_n^P \mathcal{V}_1^P - \Lambda^{-P} \mathcal{U}_1^P & \Lambda^{-P} \mathcal{X}_n - \mathcal{X}_n^P \Lambda^{-1} \\ \mathcal{U}_1 \mathcal{V}_1^P - \mathcal{V}_1 \mathcal{U}_1^P & \mathcal{U}_1 \Lambda^{-1} + \mathcal{V}_1 \mathcal{X}_n \end{bmatrix} \\
&= \begin{bmatrix} -\mathcal{X}_n^P \mathcal{V}_1^P - \mathbf{I}_m + \mathcal{X}_n^P \mathcal{V}_1^P & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_m - \mathcal{V}_1 \mathcal{X}_n^P + \mathcal{V}_1 \mathcal{X}_n \end{bmatrix} \\
&= \begin{bmatrix} -\mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_m \end{bmatrix}.
\end{aligned}$$

From the proof of Corollary 4.2, then,

$$\begin{aligned}
\Delta_{\mathcal{X}_0^T, \mathcal{X}_0}(B) &= \begin{bmatrix} \dot{\mathcal{U}} & \dot{\mathcal{V}} \end{bmatrix} A^{-1} A \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_m \end{bmatrix} A^P A^{-P} \begin{bmatrix} \mathcal{V}^P \\ \mathcal{U}^P \end{bmatrix} \\
&= \begin{bmatrix} \mathcal{W} & \mathcal{X} \end{bmatrix} \begin{bmatrix} -\mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_m \end{bmatrix} \begin{bmatrix} \dot{\mathcal{X}}^P \\ \dot{\mathcal{W}}^P \end{bmatrix} = GDH^T,
\end{aligned}$$

and therefore by Corollary 4.1, $B = \mathfrak{G}_B(GDH^T)$.

□

Proof of Proposition 4.12. Invoking Proposition 4.4 and the definition of $\nabla^{(2)}$, it is evident that

$$\begin{aligned}
\nabla^{(2)}(T) &= \begin{bmatrix} -\nabla_{Y_0, Y_0}(A_{-1}) & \cdots & -\nabla_{Y_0, Y_0}(A_{1-n}) & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \nabla_{Y_0, Y_0}(A_{1-n}) \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \nabla_{Y_0, Y_0}(A_{-1}) \end{bmatrix} \\
&= \begin{bmatrix} e_1 h_{-1}^P - h_{-1} e_1^P & \cdots & e_1 h_{1-n}^P - h_{1-n} e_1^P & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & h_{1-n} e_1^P - e_1 h_{1-n}^P \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & h_{-1} e_1^P - e_1 h_{-1}^P \end{bmatrix} \\
&= \mathbf{h} e_1^P + e_1 \mathbf{h}^P - \begin{bmatrix} h_{-1} e_1^P & \cdots & h_{1-n} e_1^P & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & e_1 h_{1-n}^P \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & e_1 h_{-1}^P \end{bmatrix} \\
&= \mathbf{h} e_1^P + e_1 \mathbf{h}^P - (\mathbb{H} \mathcal{E}_1^P + \mathcal{E}_1 \mathbb{H}^P).
\end{aligned}$$

□

Proof of Proposition 4.11. Each point is addressed in order.

1. Noting that $X_0^P = X_0^{\mathcal{P}} = X_0^P = X_0$ (and similarly for Y_0), it follows that

$$\begin{aligned}
\left(\nabla^{(2)}(A)\right)^P &= (X_0 Y_0 A - X_0 A Y_0 - Y_0 A X_0 + A X_0 Y_0)^P \\
&= A^P Y_0 X_0 - Y_0 A^P X_0 - X_0 A^P Y_0 + Y_0 X_0 A^P.
\end{aligned}$$

Since X_0 and Y_0 commute, this implies $\left(\nabla^{(2)}(A)\right)^P = \left(\nabla^{(2)}(A)^P\right)$.

2. This point follows directly from point 1 by simply taking the blockwise and block-level persymmetric transform of each side of the equality, respectively.
3. This point follows directly from points 1 and 2.
4. Consider the blocks of the individual terms of the displacement of A :

$$(X_0 Y_0 A)_{i,j} = \begin{cases} \mathbf{0} & i = 1 \\ Z_0 A_{i-1,j} & \text{else} \end{cases} \quad (X_0 A Y_0)_{i,j} = \begin{cases} \mathbf{0} & i = 1 \\ A_{i-1,j} Z_0 & \text{else} \end{cases}$$

$$(Y_0 X_0 A)_{i,j} = \begin{cases} Z_0 A_{i,j+1} & j < n \\ \mathbf{0} & \text{else} \end{cases} \quad (A Y_0 X_0)_{i,j} = \begin{cases} Z_0 A_{i,j+1} & j < n \\ \mathbf{0} & \text{else} \end{cases}.$$

The block $(1, n)$ is equal to $\mathbf{0}$ regardless of the matrix A . The remaining blocks of the displacement for $i = 1, j < n$ are given by

$$\left[\nabla^{(2)}(A) \right]_{1,j} = A_{1,j+1} Z_0 - Z_0 A_{1,j+1} = -\nabla_{Z_0, Z_0}(A_{1,j+1}),$$

the blocks $i > 1$ and $j = n$ by

$$\left[\nabla^{(2)}(A) \right]_{i,n} = Z_0 A_{i-1,n} - A_{i-1,n} Z_0 = \nabla_{Z_0, Z_0}(A_{i-1,n}),$$

and the blocks $i > 1$ and $j < n$ by

$$\left[\nabla^{(2)}(A) \right]_{i,j} = Z_0 A_{i-1,j} + A_{i,j+1} Z_0 - A_{i-1,j} Z_0 - Z_0 A_{i,j+1} = \nabla_{Z_0, Z_0}(A_{i-1,j}) - \nabla_{Z_0, Z_0}(A_{i,j+1}).$$

First consider the case where A is blockwise persymmetric. Under this assumption,

$A_{i,j}^P = A_{i,j}$, and therefore

$$\nabla_{Z_0, Z_0}(A_{i,j}) + \left(\nabla_{Z_0, Z_0}(A_{i,j}) \right)^P = Z_0 A_{i,j} - A_{i,j} Z_0 + A_{i,j} Z_0 - Z_0 A_{i,j} = \mathbf{0}.$$

As a consequence, each block of the $\nabla^{(2)}(A) + \left(\nabla^{(2)}(A) \right)^{\mathcal{P}}$ is equal to $\mathbf{0}$, and therefore

$$\nabla^{(2)}(A) = -\left(\nabla^{(2)}(A) \right)^{\mathcal{P}}.$$

Next, when A is block-level persymmetric $A_{i,j} = A_{n+1-j,n+1-i}$. Then for $i = 1$ and $j < n$,

$$\begin{aligned} \left[\nabla^{(2)}(A) + \left(\nabla^{(2)}(A) \right)^{\mathcal{P}} \right]_{1,j} &= \left[\nabla^{(2)}(A) \right]_{1,j} + \left[\nabla^{(2)}(A) \right]_{n+1-j,n} \\ &= -\nabla_{Z_0, Z_0}(A_{1,j+1}) + \nabla_{Z_0, Z_0}(A_{n-j,n}) \\ &= \nabla_{Z_0, Z_0}(A_{1,j+1} - A_{n-j,n}) = \nabla_{Z_0, Z_0}(A_{1,j+1} - A_{1,j+1}) = \mathbf{0}. \end{aligned}$$

Similarly, for $j = n$ and $i > 1$

$$\begin{aligned} \left[\nabla^{(2)}(A) + \left(\nabla^{(2)}(A) \right)^{\mathcal{P}} \right]_{i,n} &= \left[\nabla^{(2)}(A) \right]_{i,n} + \left[\nabla^{(2)}(A) \right]_{1,n+1-i} \\ &= -\nabla_{Z_0, Z_0}(A_{i-1,n}) + \nabla_{Z_0, Z_0}(A_{1,n+2-i}) \\ &= \nabla_{Z_0, Z_0}(A_{i-1,n} - A_{1,n+2-i}) = \nabla_{Z_0, Z_0}(A_{i-1,n} - A_{i-1,n}) = \mathbf{0} \end{aligned}$$

and for $i > 1$ and $j < n$

$$\begin{aligned} \left[\nabla^{(2)}(A) + \left(\nabla^{(2)}(A) \right)^{\mathcal{P}} \right]_{i,j} &= \left[\nabla^{(2)}(A) \right]_{i,j} + \left[\nabla^{(2)}(A) \right]_{n+1-j,n+1-i} \\ &= \nabla_{Z_0, Z_0}(A_{i-1,j}) - \nabla_{Z_0, Z_0}(A_{i,j+1}) \\ &\quad + \nabla_{Z_0, Z_0}(A_{n-j,n+1-i}) - \nabla_{Z_0, Z_0}(A_{n+1-j,n+2-i}) \\ &= \nabla_{Z_0, Z_0}(A_{i-1,j} - A_{i,j+1} + A_{n-j,n+1-i} - A_{n+1-j,n+2-i}) \\ &= \nabla_{Z_0, Z_0}(A_{i-1,j} - A_{i,j+1} + A_{i,j+1} - A_{i-1,j}) = \mathbf{0}. \end{aligned}$$

Therefore, $\nabla^{(2)}(A) = -\left(\nabla^{(2)}(A) \right)^{\mathcal{P}}$. The remaining equality follows from point 1. \square

Proof of Theorem 4.6. Theorem 4.5 establishes that the matrix

$$\Phi = \mathcal{F}_{-1}^T T \mathcal{F}_{+1}^H = (\mathbf{F}_{-1}^{(n)} \otimes \mathbf{I}_m)^H T (\mathbf{F}_{+1}^{(n)} \otimes \mathbf{I}_m)$$

is an $mn \times mn$ generalized block-Cauchy matrix with blocks

$$\Phi_{i,j} = U_i^T V_j (\omega_{i-1}^- - \omega_{j-1}^+)^{-1},$$

where U , V , G , and H are defined as in the theorem. Denote the block columns of G and H as

$$G_1 = \begin{bmatrix} A_0 \\ A_1 - A_{1-n} \\ \vdots \\ A_{n-1} - A_{-1} \end{bmatrix}, \quad G_2 = \begin{bmatrix} -\mathbf{I}_m \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \quad H_1 = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \\ -\mathbf{I}_m \end{bmatrix}, \quad \text{and} \quad H_2 = \begin{bmatrix} (A_{n-1} + A_{-1})^T \\ \vdots \\ (A_1 + A_{1-n})^T \\ A_0^T \end{bmatrix},$$

and their transformations as

$$\mathcal{F}_{-1}G_2 = \mathcal{F}_{-1} \begin{bmatrix} -\mathbf{I}_m \\ 0 \\ \vdots \\ 0 \end{bmatrix} = -\frac{1}{\sqrt{n}} (\mathbf{1}_n \otimes \mathbf{I}_m), \quad Q = \mathcal{F}_{-1}G_1 = \begin{bmatrix} Q_1 \\ \vdots \\ Q_n \end{bmatrix},$$

$$\overline{\mathcal{F}}_{+1}H_1 = \overline{\mathcal{F}}_{+1} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -\mathbf{I}_m \end{bmatrix} = -\frac{1}{\sqrt{n}} (\omega^+ \otimes \mathbf{I}_m), \quad \text{and} \quad P^T = \overline{\mathcal{F}}_{+1}H_2 = \begin{bmatrix} P_1^T \\ \vdots \\ P_n^T \end{bmatrix},$$

where $\omega^+ = [\omega_i^+]$. The block numerators are then

$$U_i^T V_j = -\frac{1}{\sqrt{n}} \begin{bmatrix} Q_i & \mathbf{I}_m \end{bmatrix} \begin{bmatrix} \omega_{j-1}^+ \mathbf{I}_m \\ P_j \end{bmatrix} = -\frac{1}{\sqrt{n}} (Q_i \omega_{j-1}^+ + P_j).$$

As a consequence, the blocks of Φ are

$$\begin{aligned} \Phi_{i,j} &= -\frac{1}{\sqrt{n}} (Q_i \omega_{j-1}^+ + P_j) (\omega_{i-1}^- - \omega_{j-1}^+)^{-1} \\ &= -\frac{1}{\sqrt{n}} (Q_i + P_j \omega_{n+1-j}^+) (\omega_{i-1}^- - \omega_{j-1}^+)^{-1} \omega_{j-1}^+. \end{aligned}$$

One may then consider the individual terms of this expression. First,

$$\begin{aligned} Q_i &= (\mathcal{F}_{-1}G_1)_i = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} (\omega_{i-1}^-)^k (A_k - A_{k-n}) \\ &= \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} (\omega_{i-1}^+)^k \eta^k (A_k - A_{k-n}), \end{aligned}$$

where it is assumed that $A_{-n} = \mathbf{0}$. Second,

$$\begin{aligned}
P_j &= (H_2^T \mathcal{F}_{+1}^H)_j = (H_2^T \overline{F_{+1}})_j = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} (A_{n-1-k} + A_{-1-k}) \overline{(\omega_k^+)^{j-1}} \\
&= \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} (A_{n-1-k} + A_{-1-k}) (\omega_k^+)^{n+1-j} = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} (A_{n-1-k} + A_{-1-k}) (\omega_{j-1}^+)^{n-k} \\
&= \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} (A_k + A_{k-n}) (\omega_{j-1}^+)^{k+1}
\end{aligned}$$

Putting these developments together, the individual numerator terms become

$$\begin{aligned}
(Q_i + P_j \omega_{n-j+1}^+) &= \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} A_k \left[(\omega_{j-1}^+)^{k+1} (\omega_{n-j+1}^+) + \eta^k (\omega_{i-1}^+)^k \right] + \\
&\quad A_{k-n} \left[(\omega_{j-1}^+)^{k+1} \omega_{n-j+1}^+ - \eta^k (\omega_{i-1}^+)^k \right] \\
&= \frac{1}{\sqrt{n}} \sum_{k=1-n}^{n-1} \gamma_{i,j}^{(k)} A_k,
\end{aligned}$$

where after simplification it can be shown that

$$\gamma_{i,j}^{(k)} = (\omega_{j-1}^+)^k + (\omega_{i-1}^+)^k \eta^k = (\omega_{j-1}^+)^k + (\omega_{i-1}^-)^k.$$

Since each term $\Psi_{i,j} = (Q_i + P_j \omega_{n-j+1}^+)$ is a linear combination of Toeplitz matrices, it is itself Toeplitz. Let the coefficients of each Toeplitz matrix A_k be $a_{k,\ell}$ for $\ell = (1 - m), \dots, (m - 1)$. Then

$$\mathbf{F}_{-1}^{(m)} A_k (\mathbf{F}_{+1}^{(m)})^H = \Xi_k, \quad (107)$$

for $k = (1 - n), \dots, (n - 1)$, is a Cauchy matrix of the form

$$(\Xi_k)_{\ell,r} = \frac{u_{k,\ell}^T v_{k,r}}{\phi_{\ell-1}^+ - \phi_{r-1}^-}.$$

This is an exact analog of the result Heinig proved in [53] for scalar Toeplitz matrices, and the definitions of $u_{k,\ell}$ and $v_{k,r}$ follow the block case developed in Section 4.3.4.2.

Defining a matrix

$$\widetilde{\Phi} = (\mathbf{I}_n \otimes (\mathbf{F}_{-1}^{(m)})^H) \Phi (\mathbf{I}_n \otimes \mathbf{F}_{+1}^{(m)}), \quad (108)$$

then

$$\widetilde{\Phi}_{i,j} = \frac{1}{n}(\omega_{i-1}^+ - \omega_{j-1}^-)^{-1} \omega_{j-1}^+ \sum_{k=1-n}^{n-1} \gamma_{i,j}^{(k)} \mathcal{F}_{-1}^H A_k \mathcal{F}_{+1} = \frac{1}{n}(\omega_{i-1}^+ - \omega_{j-1}^-)^{-1} \omega_{j-1}^+ \sum_{k=1-n}^{n-1} \gamma_{i,j}^{(k)} \Xi_k.$$

The system $\Phi x = y$ can thus be replaced by $\widetilde{\Phi}(\mathbf{I}_n \otimes \mathbf{F}_{+1}^{(m)})^H x = (\mathbf{I}_n \otimes \mathbf{F}_{-1}^{(m)})y$. For the remainder of the proof, systems of the form $\widetilde{\Phi}x = y$ are considered.

Using the definitions of the various quantities, the entries of $\widetilde{\Phi}$ are given by

$$[\widetilde{\Phi}_{i,j}]_{\ell,r} = -\frac{1}{n} \left(\frac{1}{\omega_{i-1}^- - \omega_{j-1}^+} \right) \omega_{j-1}^+ \left(\frac{1}{\phi_{\ell-1}^- - \phi_{r-1}^+} \right) \left[\sum_{k=1-n}^{n-1} \gamma_{i,j}^{(k)} z_{k,\ell}^T v_{k,r} \right].$$

What remains necessary is a useful description of the coefficients

$$\sum_{k=1-n}^{n-1} \gamma_{i,j}^{(k)} u_{k,\ell}^T v_{k,r}.$$

The submatrices g_k and h_k are given by

$$g_k = \begin{bmatrix} a_{k,0} & -1 \\ (a_{k,1} - a_{k,1-m}) & 0 \\ \vdots & \vdots \\ (a_{k,m-1} - a_{k,1}) & 0 \end{bmatrix} \quad \text{and} \quad h_k = \begin{bmatrix} 0 & (a_{k,m-1} + a_{k,-1}) \\ \vdots & \vdots \\ 0 & (a_{k,1} + a_{k,1-m}) \\ -1 & (a_{k,0}) \end{bmatrix}.$$

Thus, much as in the block case,

$$u_{k,\ell}^T v_{k,r} = -\frac{1}{m}(q_{k,\ell} \phi_{r-1}^+ + p_{k,r}) = -\frac{1}{m} \sum_{s=1-m}^{m-1} \mu_{\ell,r}^{(s)} a_{k,s},$$

where $q_k = \mathcal{F}_{-1} g_{k,1}$ (the first column of g_k), $p_k = \overline{\mathcal{F}_{+1}} h_{k,2}$ (the second column of h_k), and

$$\mu_{\ell,r}^{(s)} = (\phi_{\ell-1}^-)^s + (\phi_{r-1}^+)^s.$$

Putting the various pieces together yields

$$[\widetilde{\Phi}_{i,j}]_{\ell,r} = \frac{1}{mn} \left(\frac{\omega_{j-1}^+}{\omega_{i-1}^- - \omega_{j-1}^+} \right) \left(\frac{\phi_{r-1}^+}{\phi_{\ell-1}^- - \phi_{r-1}^+} \right) \sum_{k=1-n}^{n-1} \gamma_{i,j}^{(k)} \sum_{s=1-m}^{m-1} \mu_{\ell,r}^{(s)} a_{k,s}. \quad (109)$$

The main statement of the proof may now be confirmed. Multiplying out the expansion term yields

$$\begin{aligned} \gamma_{i,j}^{(k)} \mu_{\ell,r}^{(s)} &= \left(((\omega_{i-1}^-)^k + (\omega_{j-1}^+)^k) \left((\phi_{\ell-1}^-)^s + (\phi_{r-1}^+)^s \right) \right) \\ &= (\omega_{i-1}^-)^k (\phi_{\ell-1}^-)^s + (\omega_{i-1}^-)^k (\phi_{r-1}^+)^s + (\omega_{j-1}^+)^k (\phi_{\ell-1}^-)^s + (\omega_{j-1}^+)^k (\phi_{r-1}^+)^s. \end{aligned}$$

From this expression, it is evident that

$$\sum_{k=1-n}^{n-1} \gamma_{i,j}^{(k)} \sum_{s=1-m}^{m-1} \mu_{\ell,r}^{(s)} a_{k,s} = mn \left([\Theta_{00}]_{i,\ell} - [\Theta_{01}]_{i,r} - [\Theta_{10}]_{j,\ell} + [\Theta_{11}]_{j,r} \right).$$

By (109), then,

$$[\tilde{\Phi}_{i,j}]_{\ell,r} = \frac{[\Theta_{00}]_{i,\ell} - [\Theta_{01}]_{i,r} - [\Theta_{10}]_{j,\ell} + [\Theta_{11}]_{j,r}}{(\omega_{i-1}^- - \omega_{j-1}^+)(\phi_{\ell-1}^- - \phi_{r-1}^+)} \omega_{j-1}^+ \phi_{r-1}^+.$$

Using the definitions of (105), (106), and Δ , the theorem is proved. □

APPENDIX B

TANGENTIAL-INTERPOLATION EXAMPLE

To demonstrate the divide-and-conquer basis construction, consider the problem of interpolating two degree-1 polynomials $x(z) = x_0 + x_1z$ and $y(z) = y_0 + y_1z$ that obey the four interpolation conditions

$$\begin{aligned} x(\omega_0) - y(\omega_0) - 2 &= 0, & (1 - \mathbf{j})y(\omega_1) + \mathbf{j}2 &= 0, \\ x(\omega_2) + y(\omega_2) - 2 &= 0, & x(\omega_3) + \mathbf{j}y(\omega_3) &= 0, \end{aligned}$$

where $\omega_k = \mathbf{e}^{j2\pi k/4}$. These conditions can be expressed as a tangential-interpolation problem by gathering $x(z)$ and $y(z)$ into the polynomial vector $\mathbf{p}^*(z) = \begin{bmatrix} x(z) & y(z) & 1 \end{bmatrix}^T$, defining $\tau = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T$, and setting

$$\Lambda_{11} = \text{diag}(1, 0, 1, 1) \quad \Lambda_{12} = \text{diag}(-1, (1 - \mathbf{j}), 1, \mathbf{j}) \quad \Lambda_{13} = \text{diag}(-2, \mathbf{j}2, -2, 0).$$

Using these definitions, the interpolation conditions correspond to the linear system

$$\begin{bmatrix} \Lambda_{11}\mathcal{F}_2 & \Lambda_{12}\mathcal{F}_2 & \Lambda_{13}\mathcal{F}_1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{0},$$

which in turn corresponds to the set of tangential-interpolation conditions

$$\begin{bmatrix} \lambda_{11}^{(k)} & \lambda_{12}^{(k)} & \lambda_{13}^{(k)} \end{bmatrix} \mathbf{p}^*(\omega_k) = \phi_k \mathbf{p}^*(\omega_k) = 0.$$

The example proceeds with interleaving data splitting, and after the first subdivision of the data the algorithm builds a basis using the interpolation conditions $\{\{\phi_0, \omega_0\}, \{\phi_2, \omega_2\}\}$. Once this basis is determined, the algorithm computes a second basis using the interpolation conditions $\{\{\phi_1, \omega_1\}, \{\phi_3, \omega_3\}\}$.

To start the construction of the first basis, it is initialized to the identity matrix, which has τ -degrees $\delta_1 = \delta_2 = -1$ and $\delta_3 = 0$. Since the first and second columns have the same τ -degree, the column corresponding to the interpolation condition of largest magnitude is the column whose τ -degree will be increased. Since these conditions have the same magnitude, the first column is chosen by default, and the basis constructed from the first point is

$$\mathbf{B}_1(z) = \begin{bmatrix} (z-1) & 1 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

This basis satisfies $\phi_0^T \mathbf{B}_1(\omega_0) = \mathbf{0}$.

The τ -degrees of $\mathbf{B}_1(z)$ are now $\{0, -1, 0\}$, and so to process the interpolation data $\{\phi_2, \omega_2\}$, it is necessary to construct a $\tilde{\tau}$ -reduced basis $\mathbf{B}_{1 \rightarrow 2}(z)$, where $\tilde{\tau} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$, for the interpolation condition

$$\tilde{\phi}_2^T = \phi_2^T \mathbf{B}_1(\omega_2) = \begin{bmatrix} 1 & 1 & -2 \end{bmatrix} \begin{bmatrix} -2 & 1 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -2 & 2 & 0 \end{bmatrix}.$$

The evaluation of $\mathbf{B}_1(\omega_2)$ is straightforward since the degrees of the elements of $\mathbf{B}_1(z)$ are small. However, these type of evaluations are computed with the FFT in practice.

Initializing the new construction with the identity, its second column has the lowest $\tilde{\tau}$ -degree. The resulting basis is then

$$\mathbf{B}_{1 \rightarrow 2}(z) = \begin{bmatrix} 1 & 0 & 0 \\ 1 & (z+1) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

for the second interpolation condition. With these two bases computed, they are next multiplied together:

$$\mathbf{B}_2(z) = \mathbf{B}_1(z) \mathbf{B}_{1 \rightarrow 2}(z) = \begin{bmatrix} z & (z+1) & 2 \\ 1 & (z+1) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

For larger problems, the basis multiplication is computed using FFTs to calculate the polynomial products. The resulting basis $\mathbf{B}_2(z)$ now satisfies $\phi_0^T \mathbf{B}_2(\omega_0) = \phi_2^T \mathbf{B}_2(\omega_2) = \mathbf{0}$.

There remain the two interpolation points $\{\{\phi_1, \omega_1\}, \{\phi_3, \omega_3\}\}$. These points are used to compute a second basis $\mathbf{B}_4(z)$ such that the product $\mathbf{B}_2(z)\mathbf{B}_4(z)$ satisfies all of the interpolation conditions. The first step in computing $\mathbf{B}_4(z)$ is to update the remaining interpolation conditions:

$$\hat{\phi}_1^T = \phi_1^T \mathbf{B}_2(\omega_1) = \begin{bmatrix} 0 & (1 - \mathbf{j}) & \mathbf{j}2 \end{bmatrix} \begin{bmatrix} \mathbf{j} & (1 + \mathbf{j}) & 2 \\ 1 & (1 + \mathbf{j}) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} (1 - \mathbf{j}) & 2 & \mathbf{j}2 \end{bmatrix}$$

and

$$\hat{\phi}_3^T = \phi_3^T \mathbf{B}_2(\omega_3) = \begin{bmatrix} 1 & \mathbf{j} & 0 \end{bmatrix} \begin{bmatrix} -\mathbf{j} & (1 - \mathbf{j}) & 2 \\ 1 & (1 - \mathbf{j}) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 2 \end{bmatrix}.$$

Since the τ -degree of each column of $\mathbf{B}_2(z)$ is zero, $\mathbf{B}_4(z)$ will be a $\tilde{\tau}$ -reduced basis, where $\tilde{\tau} = \mathbf{0}$. Starting with the condition $\{\hat{\phi}_1, \omega_1\}$ and the identity matrix, all columns have equal $\tilde{\tau}$ -degree, and the column of $\hat{\phi}_1$ with largest magnitude is again selected. This yields the basis

$$\mathbf{B}_3(z) = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2}(1 - \mathbf{j}) & (z - \mathbf{j}) & -\mathbf{j} \\ 0 & 0 & 1 \end{bmatrix}.$$

To process the last remaining condition, it must first be updated:

$$\tilde{\phi}_3^T = \hat{\phi}_3^T \mathbf{B}_3(\omega_3) = \begin{bmatrix} 0 & 2 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2}(1 - \mathbf{j}) & -\mathbf{j}2 & -\mathbf{j} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} (-1 + \mathbf{j}) & -\mathbf{j}4 & (2 - \mathbf{j}2) \end{bmatrix}.$$

The $\tilde{\tau}$ -degrees of $\mathbf{B}_3(z)$ are $\{0, 1, 0\}$, and therefore a basis $\mathbf{B}_{3 \rightarrow 4}(z)$ is constructed to increase the $\tilde{\tau}$ -degree of either the first or third column of $\mathbf{B}_3(z)$. Again choosing the condition with

largest magnitude,

$$\mathbf{B}_{3 \rightarrow 4}(z) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2} & (-1 + \mathbf{j}) & (z + \mathbf{j}) \end{bmatrix}.$$

The matrix $\mathbf{B}_4(z)$ is then computed as the product

$$\mathbf{B}_4(z) = \mathbf{B}_3(z)\mathbf{B}_{3 \rightarrow 4}(z) = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & (z + 1) & -\mathbf{j}z + 1 \\ \frac{1}{2} & -1 + \mathbf{j} & z + \mathbf{j} \end{bmatrix}.$$

With each half of the interpolation problem solved, the final basis is the result of one last multiplication:

$$\mathbf{B}(z) = \mathbf{B}_2(z)\mathbf{B}_4(z) = \begin{bmatrix} \frac{1}{2}(z + 1) & z^2 + z + (-1 + \mathbf{j}2) & -\mathbf{j}z^2 + (3 - \mathbf{j})z + (1 + \mathbf{j}2) \\ -\frac{1}{2}(z - 1) & z^2 + 2z + 1 & -\mathbf{j}z^2 + (1 - \mathbf{j})z + 1 \\ \frac{1}{2} & (-1 + \mathbf{j}) & z + \mathbf{j} \end{bmatrix}.$$

In the final basis, the first column has the desired degree structure. Scaling this column so that its last entry is 1 yields

$$\hat{x}(z) = z + 1 \quad \hat{y}(z) = 1 - z.$$

It can then be verified that these polynomials satisfy the interpolation conditions:

$$x(\omega_0) - y(\omega_0) - 2 = 2 - 0 - 2 = 0$$

$$(1 - \mathbf{j})y(\omega_1) + \mathbf{j}2 = (1 - \mathbf{j})(1 - \mathbf{j}) + \mathbf{j}2 = 0$$

$$x(\omega_2) + y(\omega_2) - 2 = 0 + 2 - 2 = 0$$

$$x(\omega_3) + \mathbf{j}y(\omega_3) = 1 - \mathbf{j} + \mathbf{j} + 1 = 0.$$

APPENDIX C

SPIRALFFT

This appendix describes a fast algorithm for computing 3-D NUFFTs of spatial volumes on spiral contours in frequency space in the context of 3-D MRI [106, 107]. Spiral sampling modalities are common to 3-D MRI, as spiral patterns allow for efficient scanning of a large range of frequencies. By parameterizing the spiral contours in a certain way, the computations required for an NUFFT can be broken down into a series of 1-D transforms. This process results in a transform that is in some ways separable while still yielding patterns that can provide a sufficiently dense coverage of frequency space.

C.1 Context

Section 3.1.5.5 provided a brief introduction to the non-uniform FFT, but omitted many details of its computation. The most well-known NUFFT algorithms are all variations on the same essential theme: an oversampled FFT is computed to densely sample the DTFT of the signal on a regular grid and the samples are then used to interpolate the spectrum at the desired frequencies [13, 29, 98, 113]. For 1- and 2-D signals, the computational cost of this process is not overwhelming when compared to the FFT. In 3-D, however, the interpolation step dominates the computation even for coarse accuracy (*i.e.* highly truncated interpolation kernels), and the total cost tends to far exceed that of a 3-D FFT for large signals. Indeed, experimental results (see [65], Table 5.2) have shown the execution times of the NUFFT and the FFT to vary by orders of magnitude in some cases.

When the signal spectrum is sampled along groups of spirals in the 3-D spatial frequency domain (referred to as “cones” in the MRI community) that satisfy certain constraints, the NUDFT can be decomposed as a series of 1-D FFTs and chirp Z-transforms

(CZTs), making the whole transform “effectively separable.” Using this sampling modality, the NUDFT can be computed fast and nearly exactly, which offers a significant improvement over general-purpose NUFFTs for a broad range of parameter regimes.

C.1.1 Motivation

The algorithm of this appendix samples the DTFT of a 3-D signal along spiral contours in the spatial frequency domain (“ k -space”). This sampling geometry is prevalent in MRI because of the convenience it offers the acquisition process. MRI scanners capable of sampling along spiral contours in k -space provide high-resolution images with modest acquisition times [2]. Spiral contours are also not as sensitive to some systematic errors in MRI when compared to other modalities as a byproduct of the structure of the magnetic gradients involved in the technology [75].

The problem at hand is specified as follows. Let k_1 , k_2 , and k_3 denote 3-D spatial frequency coordinates. The sampling patterns considered are composed of spiral contours in the spatial frequency domain. The index q is used to identify a specific spiral, while the index r is used to specify a sample along a given spiral. For a spiral aligned with the k_3 axis, for instance, the r^{th} sample along the q^{th} spiral has coordinates

$$k_1[r, q] = \alpha[r, q] \cos(\beta[r, q]) \quad k_2[r, q] = \alpha[r, q] \sin(\beta[r, q]) \quad k_3[r, q] = \rho[r, q], \quad (110)$$

where ρ defines the “spiral height,” α defines the “spiral radius,” and β defines the “spiral angle.” Given an $N \times N \times N$ voxel volume g , the DTFT of g is sampled at point the r along spiral q by computing the NUDFT sum

$$\hat{g}[r, q] = \sum_{m=-N/2}^{N/2-1} \sum_{n=-N/2}^{N/2-1} \sum_{p=-N/2}^{N/2-1} g[m, n, p] e^{-j2\pi(k_1[r, q]m + k_2[r, q]n + k_3[r, q]p)}. \quad (111)$$

For a collection of R samples r_0, r_1, \dots, r_{R-1} , the sum (111) is evaluated at each point with the matrix-vector multiplication

$$\hat{\mathbf{g}} = \mathbf{A} \mathbf{g}, \quad \text{with} \quad A_{s,t} = \mathbf{e}^{-j2\pi \vec{\ell}[t]^T \vec{k}[s]}, \quad (112)$$

where $\vec{\ell}[t]$ is a 3-D spatial index vector in $[-N/2, N/2)^3$ and $\vec{k}[s]$ is a 3-D k -space frequency vector in $[-1/2, 1/2)^3$. The matrix A maps the uniform spatial samples $\{g[m, n, p]\}$ to spectral samples $\{\hat{g}[r, q]\}$ at the specified locations along spirals.

Of more interest than simply computing the matrix-vector product Ag is the inverse problem, where the spectral samples \hat{g} are supplied and the spatial volume g is to be reconstructed. Iterative solution methods for this problem such as CG invariably involve multiple applications of the matrices A and A^H , so if an iterative method is to be employed it is crucial to have a fast algorithm for applying A .

C.1.2 Existing methods

The NUDFT cannot be computed with the same efficiency as the FFT, but NUFFT's have been developed that *approximately* apply the matrix A in (111). The product is approximated by interpolating the values of an oversampled FFT to the off-grid locations. The general procedure consists of the following steps¹:

1. Deconvolve the input spatial/time signal with the spatial/time-domain representation of the interpolation kernel;
2. Compute an oversampled FFT of the result of step 1 to prevent aliasing effects;
3. Convolve the result of the previous step with the frequency-domain representation of the interpolation kernel; and
4. Evaluate the result at the desired off-grid frequencies.

The interpolation gives the NUFFT an inherent complexity-*vs.*-accuracy tradeoff. An interpolation of length $2c + 1$ along each dimension yields a per-sample accuracy of $\mathcal{O}(\log(1/\epsilon))$ at a per-sample cost of $\mathcal{O}((2c + 1)^d)$, where d is the dimension of the problem [88]. For a half-window length c , the cost of an NUFFT computing s total spectral samples of an input

¹In practice, the final two steps are combined by interpolating a discrete spectrum.

volume with t samples is $O(t \log t + s(2c + 1)^d)$. The first term of this cost is the cost of the FFT, while the second is the cost of the interpolation.

Various interpolation techniques have been proposed for NUFFTs, including Kaiser-Bessel kernels [58, 81], Gaussian kernels [29, 93], least-squares locally-optimized kernels [96], and min-max interpolation [37]. While for 1-D and 2-D problems the cost of the interpolation is relatively minor, for 3-D problems the interpolation cost can be large relative to the cost of the FFT. A simple computational example can be used to demonstrate this fact.

In this example, the NUFFT was computed for a pair of two signals: the first a 2-D signal with side length 75 (for a total of 5,625 spatial points), and the second is a 3-D signal with side length 14 (for a total of 2,744 spatial points). If the NUFFTs are computed with FFTs oversampled by a factor of 2, the FFT of the 2-D signal uses 22,500 samples and the FFT of the 3-D signal uses 21,952 samples. Predictably, the execution times for the FFTs are approximately equal, requiring 1.90 ms for the first and 2.32 ms for the second.²

Using the same interpolation quality and number of output frequencies (112,500), a MATLAB implementation of the NUFFT executes in 36.3 ms for the 2-D signal and 139.3 ms for the 3-D signal. This difference is nearly a full order of magnitude despite the similar problem sizes. While the absolute difference in time might not be appreciable for this example, it is demonstrative.

Several methods have been proposed to help reduce interpolation cost (see [49], for example), but they require an amount of additional storage that can be burdensome or even impossible for large-scale problems. In the end, the interpolation cost can be shifted to the front-end of the computation, but it cannot be totally avoided. Unfortunately, it can make NUFFT algorithms costly and, in some cases, infeasible in 3-D.

²The experimental setup up is the same here as described in Section C.5.

C.2 The SpiralFFT

The FFT is efficient because it makes use of structure of the uniform spacing in the nodes of the Vandermonde DFT matrix. Spiral sampling patterns do not share the same properties as uniform patterns, but they can contain exploitable structure under certain conditions. The next few sections describe the SpiralFFT, a fast transform for certain types of spiral sampling patterns, with respect to spirals aligned with the k_3 axis. It is straightforward to adapt the algorithm and its analysis to other alignments, however.

C.2.1 Constrained sampling patterns

One of the underlying assumptions of SpiralFFT is that the Fourier samples are to be computed for multiple spirals. This is typically the case for spiral MRI, and the assumption is therefore not extraordinarily restrictive. To be more specific, it is assumed that the NUFT sum (111) is to be computed along Q spirals and at R locations per spiral.

The transform depends on the following discrete parameterization for the r^{th} sample along the q^{th} spiral:

$$\begin{aligned} k_1[r, q] &= q\alpha[r] \cos(\beta[r]) \\ k_2[r, q] &= q\alpha[r] \sin(\beta[r]) \\ k_3[r, q] &= \zeta[r], \end{aligned} \tag{113}$$

where α and ζ are some unspecified functions of r . The index r is referred to as the **spiral sample index** and q as the **spiral number**. There are no restrictions on the function α and ζ save that the frequency sample locations k_x must each satisfy

$$-\frac{1}{2} \leq k_x[r, q] < \frac{1}{2}.$$

While it is not strictly necessary, α and ζ are typically monotonic. The spiral angles $\beta[r]$ as prescribed as follows: for each r , there exist some $\omega[r] \in \mathbb{R}$ and $L_1[r], L_2[r] \in \mathbb{Z}$ such that

$$\begin{aligned} \cos(\beta[r]) &= \omega[r]L_1[r] \\ \sin(\beta[r]) &= \omega[r]L_2[r]. \end{aligned} \tag{114}$$

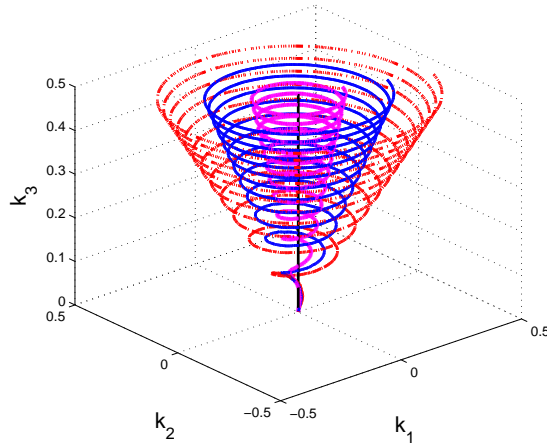


Figure 18: Examples of spirals parameterized as in (113). For this parametrization, the spirals grow as \sqrt{r} along the k_3 axis as the spiral sample index r increases. As the spiral number q increases, the radius in the (k_1, k_2) plane increases.

This choice of angles is explored further in Section C.2.3.

An example of a set of spirals fitting (113) is illustrated in Figure 18. The conditions of (113) and (114) have four important properties:

1. the spiral height depends *only* on the spiral sample index r , and is *fixed* for all spiral numbers q ;
2. the radius in the (k_1, k_2) plane grows in magnitude *linearly* with q (but may also follow the arbitrary function α across r);
3. the angle in the (k_1, k_2) plane is a *only* a function of r ; and
4. the line between any frequency pair $(k_1[r, q], k_2[r, q])$ and the origin of the (k_1, k_2) plane has rational slope.

C.2.2 Calculations in 1-D

The constraints on the spiral parameterization in the previous section allow the 3-D NUDFT of (111) to be reduced to a series of 1-D transforms. Denote $W = e^{-j2\pi}$; then the r^{th} spatial

frequency sample along the q^{th} spiral (111) becomes

$$\hat{g}[r, q] = \sum_{m,n,p} g[m, n, p] W^{(mk_1[r,q]+nk_2[r,q]+pk_3[r,q])}, \quad (115)$$

where $-\frac{N}{2} \leq m, n, p < \frac{N}{2}$. Using the parametrizations of (113), the sum in (115) can be written as

$$\hat{g}[r, q] = \sum_{m,n} W^{(mk_1[r,q]+nk_2[r,q])} \sum_p g[m, n, p] W^{\zeta[r]p}. \quad (116)$$

In (116), for a fixed pair (m, n) the inner sum is a 1-D NUFFT in r . If $\zeta[r] = \zeta_0 r$, the radius in k_3 grows linearly and the inner sum is actually a chirp-Z Transform (CZT) [89]. The CZT can be computed exactly at roughly twice the cost of an FFT — this would make the SpiralFFT exact with the extra constraint on ζ . However, it is typically faster to use a 1-D NUFFT to approximate the CZT at an acceptable accuracy.

For the remainder, it is assumed that $\zeta[r]$ is arbitrary. A series of 1-D NUFFT's can be used to transform the signal along the dimension indexed by p , yielding the intermediate signal

$$h[m, n, r] = \sum_p g[m, n, p] W^{\zeta[r]p} \quad (117)$$

for all m and n . The cost of these N^2 individual length- R NUFFT's is then

$$O(N^2(\sigma N \log(\sigma N) + (2c + 1)R)),$$

where σ is the FFT oversampling factor NUFFT and c is the interpolation window cutoff (see the description in [65]). Substituting in the equations for $k_1[r, q]$ and $k_2[r, q]$ into (117) reduces (116) to

$$\hat{g}[r, q] = \sum_{m,n} h[m, n, r] W^{q\alpha[r]\omega[r](L_1[r]m+L_2[r]n)}. \quad (118)$$

The quantities $L_1[r]$, $L_2[r]$, m , and n in (118) are all integers. Therefore, $i = L_1[r]m + L_2[r]n$ is also an integer, and is restricted to the range

$$-\frac{N}{2}L \leq i \leq \frac{N}{2}L, \quad (119)$$

where $L = |L_1[r]| + |L_2[r]|$. For each fixed value of r , the 1-D signal $h_1[i, r]$ is computed over i as

$$h_1[i, r] = \sum_{m,n:L_1[r]m+L_2[r]n=i} h[m, n, r]. \quad (120)$$

Using (120), the equation (118) can be expressed as a 1-D CZT in the variable q :

$$\hat{g}[r, q] = \sum_{i=-\frac{N}{2}L}^{\frac{N}{2}L} h_1[i, r] W^{q\alpha[r]\omega[r]i}.$$

For each r , then, the 1-D signal in (120) is computed with line sums and a CZT is applied to obtain $\hat{g}[r, q]$ for all q . Denoting the average value of L over the entire range of r as \bar{L} , the total cost of these CZTs is $O(R(N\bar{L} + Q) \log(N\bar{L} + Q))$. The quantity \bar{L} depends on how $L_1[r]$ and $L_2[r]$ are chosen, which is an issue that is addressed in detail in the ensuing section.

The computation of the NUDFT in SpiralFFT can thus be summarized as follows:

1. Choose the parameters $\alpha[r]$, $\zeta[r]$, and $\beta[r]$.
2. For each index pair (m, n) , compute a 1-D length- R NUFFT of $g[m, n, p]$ across p as prescribed by (117) and store the result in the 3-D array $h[m, n, r]$.
3. For each r :
 - (a) Compute the 1-D signal length.
 - (b) Construct the 1-D signal $h_1[i, r]$.
 - (c) Compute the 1-D length- Q CZT of $h_1[i, r]$ with CZT constant $W^{\alpha[r]\omega[r]}$, and store the result in the 2-D array $\hat{g}[r, q]$.

Pseudocode for this procedure is given in Algorithm 11.

C.2.3 Choosing the angles β

This section explores how the angles $\beta[r]$ may be chosen to satisfy the constraint (114) while also allowing nearly uniform coverage of k -space. The choice of β has a direct

Algorithm 11 Compute SpiralFFT $\hat{g}[r, q]$ of input signal $g[m, n, p]$.

procedure $\hat{g} = \text{SPIRALFFT}(g)$

Select α, ζ, β

for $(m, n) \in [-N/2, N/2]^2$ **do**

$$h[m, n, r] \leftarrow \sum_{p=-N/2}^{N/2-1} g[m, n, p] W^{\zeta[r]p} \text{ (1-D NUFFT)}$$

end for

for $r = 0, \dots, R - 1$ **do**

$$I_1 \leftarrow |L_1[r]|N + |L_2[r]|N + 2$$

for $i = -I_1/2, \dots, I_1/2 - 1$ **do**

$$h_1[i, r] \leftarrow \sum_{m,n : L_1m + L_2n = i} h[m, n, r]$$

end for

$$W_r \leftarrow W^{\alpha[r]\omega[r]}$$

$$\hat{g}[r, q] \leftarrow \sum_{i=-I_1/2}^{I_1/2-1} h_1[i, r] W_r^{iq} \text{ (1-D CZT)}$$

end for

end procedure

impact on the efficiency of the transform, as the length of the 1-D signal $h_1[i, r]$ in (119) depends on $L_1[r]$ and $L_2[r]$, which in turn are determined by $\beta[r]$. The precise relationship between these parameters is

$$\beta[r] = \tan^{-1} \left(\frac{L_2[r]}{L_1[r]} \right). \quad (121)$$

As $L_1[r]$ and $L_2[r]$ determine the length of the CZTs taken in the second step of SpiralFFT, it is desirable to select $\beta[r]$ such that $L_1[r]$ and $L_2[r]$ are as small as possible while still providing a sufficient coverage of k -space.

Let the frequency plane be divided into octants, and for simplicity consider only the first octant of angles $\beta[r] \in \left[0, \frac{\pi}{4}\right)$. The extension of the angle choices to other octants is straightforward, requiring only some basic trigonometric manipulations. The objective is to select T angles in the first octant of the frequency plane that satisfy (114). These selections are made by choosing appropriate values of $L_1[r]$ and $L_2[r]$ and solving for the corresponding $\beta[r]$ in (121).

Since only angles in the first octant are considered, the ratio L_2/L_1 is bounded from below by 0 and from above by 1. Therefore, it is necessary to select integer values for $L_1[r]$ and $L_2[r]$ that will cause their ratio to increase from 0 to 1. One simple choice (as taken in [106]) is to hold the denominator $L_1[r]$ constant, setting

$$\begin{aligned} L_1[r] &= T \\ L_2[r] &= r \end{aligned} \quad (122)$$

for $r = 0, \dots, T - 1$. With this approach, and with the natural extension to the remaining seven octants, the angles chosen for the spirals have the fairly uniform distribution illustrated in Figure 19(a). However, this approach produces large values of L_1 and L_2 , which in turn cause the lengths of some CZTs to be longer than necessary.

This problem can be avoided by selecting a set of rational numbers in the interval $[0, 1)$ that have a small average denominator. Specifically, one can use sequences of completely reduced fractions in this interval that have maximum denominator n , which are known as

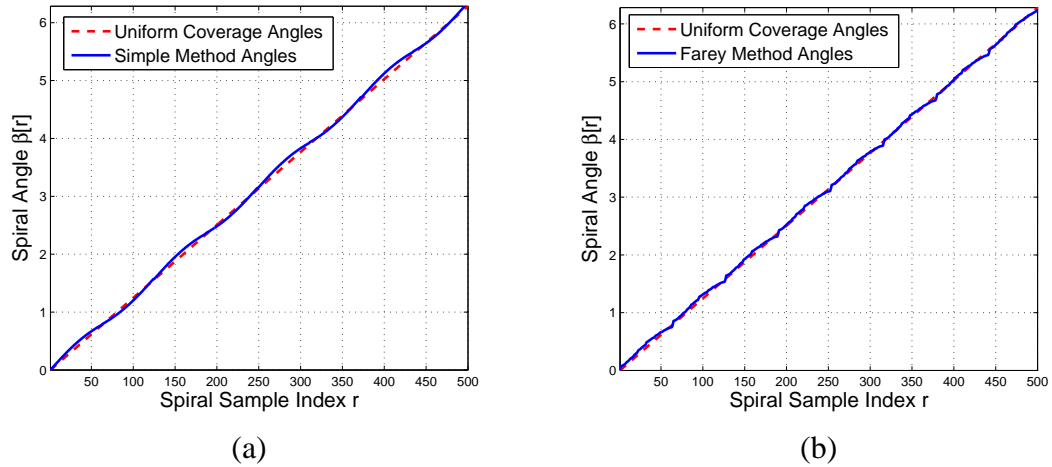


Figure 19: Parameterization angles defining rational slopes found through (a) the simple method and (b) the Farey sequence method. These angles are compared to angles uniformly covering the interval $[0, 2\pi)$. The parameterization angles do not uniformly cover the interval in the sense that they underrepresent some areas, but neither method's angles leave any significantly large gaps. Uniform angular coverage is desirable to prevent large portions of the frequency plane from being ignored in the measurements.

Farey sequences. For example, the 5th order Farey sequence is

$$\text{farey}(5) = \left\{ 0, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, 1 \right\}.$$

This sequence has 10 candidates for ratios that will produce angles in the first octant with a maximum denominator of 5.³ If T were 10, and $L_2[r]$ were chosen to be the numerators and $L_1[r]$ the denominators of the first 10 elements of the 5th order Farey sequence (setting $L_1[0] = 1$), then the average value of $L_1[r] + L_2[r]$ is 5.5. Compare this result to the method of (122) with $T = 10$, which yields a comparable set of angles but results in an average value of 14.5.

Using the entire Farey sequence to select the values of L_1 and L_2 has two drawbacks. First, for most values of T there will be more elements of the Farey sequence than are necessary, and thus some subsampling of the sequence will be necessary. Second, while the Farey sequence is relatively uniform through most of the range, it under-represents the edges of the interval 0 to 1.

³The angle $\frac{\pi}{4}$ is not considered to be in the octant, so the entry 1 is ignored.

Table 11: Average signal size for the simple and Farey methods of selecting angles yielding rational slope.

Number of Angles T	Farey Order U	Average Length (Farey)	Average Length (Simple)
5	4	4.2	7
10	5	5.5	14.5
15	7	7.3	22
20	8	8.4	29.5
25	9	9.4	37
30	10	10.2	44.5

These shortcomings can be addressed by subsampling the Farey sequence of order $U > T$ by first dropping points in the middle of the interval while keeping most of the points towards the edges. Specifically, after generating the Farey sequence of order U , the fraction with the largest denominator U that is closest to $1/2$ is removed from the sequence until only T elements remain in the set. This process has the benefit of making the sampling more uniform while simultaneously removing the elements of the sequence that produce some of the largest values of $L_1[r] + L_2[r]$. Table 11 gives the average values of $L[r] = L_1[r] + L_2[r]$ for various values of T using this technique, and Figure 19(b) shows the corresponding distribution of the angles.

The Farey method improves the efficiency of the transform as compared to the naïve approach. The second stage of the SpiralFFT uses CZTs of the 1-D signals h_1 in Algorithm 11, and the computational cost of this transform scales with the length of the input signal. If the lengths of the input signals are given as $I_1[r]$ and the length of the transformed signal is Q , then the computational cost of each CZT is $\mathcal{O}((I_1[r] + Q) \log(I_1[r] + Q))$. Table 11 indicates that even for only 30 angles, the transform executes over 4 times faster on average when the angles are chosen by the Farey method. This improved efficiency only grows more dramatic as the number of angles increases.

C.3 Computational complexity

Using the results from Section C.2.2, the total complexity of the SpiralFFT is

$$O(N^2(\sigma N \log(\sigma N) + (2c + 1)R) + R(N\bar{L} + Q) \log(N\bar{L} + Q)), \quad (123)$$

where \bar{L} is the average length of all of the 1-D flattened signals $h_1[i, r]$, $(2c+1)$ is the window length for the NUFFT in the first half of the algorithm, σ is the FFT oversampling factor for the NUFFT, and R , N , and Q are given as in Section C.2.1. The first term in (123) comes from the N^2 NUFFT performed in the first half of the algorithm, while the second term comes from the R CZTs performed in the second half. The $N\bar{L}$ terms arise from the average length of the 1-D signals that are created and the CZT that is computed across a set of Q frequencies.

No general statement can be made comparing the complexities of the SpiralFFT and NUFFT. When the number of DTFT samples is sufficiently small compared to the spatial support of the signal (*i.e.*, $QR \ll N^3$), both algorithms will be dominated by operations of complexity $O(N^3 \log(N))$. When the number of DTFT samples is much greater than the spatial support of the signal (*i.e.*, when $QR \gg N^3$), the SpiralFFT should have a significantly lower complexity than the NUFFT for most ranges of Q and R . The reasoning for this is that the cost of the NUFFT will behave as $O((2c + 1)^3 QR)$ while the cost of the SpiralFFT will behave as $O(QR \log(Q))$, which has a smaller constant term in general (and smaller storage requirements, typically).

The choices of the number of spirals Q and the number of samples-per-spiral R determine the relative complexity of the SpiralFFT. If the product QR is fixed, as R increases relative to Q the cost of applying the SpiralFFT increases. This scenario corresponds to increasing the sampling density along individual spirals while reducing the total number of spirals. The value of N will determine whether or not the SpiralFFT executes faster than the NUFFT in these circumstances. For typical 3-D MRI measurements, though, QR is on the same order as N^3 , and to cover the frequency plane as uniformly as possible Q is

usually on same order as R . A value reflective of typical spiral MRI contours for the ratio of R/Q is approximately 2.5.

The transform has an accuracy-*vs.*-efficiency trade-off that comes from the NUFFT in the first round of transforms. This tradeoff turns out to be of relatively minor importance once the precomputations described in Section C.4 are incorporated. For small data sets, the transform executes very rapidly and the increases in speed that come from reducing the accuracy are very minor. For larger data sets, since the NUFFT is the same for all N^2 times it is performed, and since only 1-D transforms are performed, the interpolation matrix can be pre-computed and stored.

For a problem of standard size, the execution time is typically reduced by only 3-5% when the interpolation window size is decreased by 2 (which usually results in a loss of approximately two digits of accuracy, although the exact change in accuracy depends on the window function). As noted above, if the function $\zeta[r]$ is linear, then the NUFFT may be replaced with a CZT. This eliminates the trade-off without an appreciable difference in speed, but places an additional limitation on the spiral structure.

C.4 Precomputations for large data sets

There are a number of precomputations that can be performed to make the SpiralFFT more efficient for large data sets and repeated applications. The first step of the transform consists of N^2 NUFFTs. Since each NUFFT is identical and operates across a single dimension, a single interpolation matrix (as described in [65]) can be fully precomputed and stored. This allows a large amount of the computation to be offloaded the front-end of the transform, significantly reducing the cost of each NUFFT. This step alone drastically reduces the constant in the complexity for the first stage of the transform.

The computations for the second step of the transform can be reduced by performing the CZTs in the correct order. FFT algorithms such as the FFTW [41] store “wisdom” information that allows a transform of a given size to be computed more efficiently each

time it is performed. By grouping the CZTs that need to be performed based on their length, the wisdom information can be accumulated throughout the process.

Finally, if the set of all CZTs that require the same FFT length are done sequentially (which may match the sequential ordering of frequency coefficients in the output array), memory accesses will be faster as the same memory locations will be repeatedly referenced by the FFT. By carefully arranging the data order, then, the number of cache misses can be reduced to improve the execution time.

C.5 Numerical experiments

Several experiments were conducted to evaluate the efficiency of the transform in various parameter ranges. The NUFFT used for comparison is the software package NFFT3 [64], which was compiled with its standard configuration. The interpolation kernel for the NFFT3 was the default Kaiser-Bessel kernel. The SpiralFFT algorithm was coded in C++, and FFTs were computed using the FFTW3 library [41]. All experiments were performed on a 3.16 GHz Intel Core 2 Duo machine with 3.0 GB of RAM and on the Ubuntu 10.04 LTS Lucid Lynx operating system.

C.5.1 Spiral count vs. spiral sampling density

As mentioned in Section C.3, when the spatial support N^3 and the total number of frequency coefficients QR are on the same order, as the spiral sampling density R increases relative to the number of spirals Q , the cost of the SpiralFFT will increase (and therefore its speed will decrease). Using a 2.25-million point spatial resolution ($N = 130$), the product QR was held fixed at 5.52-million points, and the ratio of R/Q was varied from $\frac{1}{8}$ to 8. These parameters were chosen to mimic standard 3-D spiral MRI data sets (see [51] and [72]). Table 12 holds the SpiralFFT and NUFFT transform times. Also included in Table 12 is the accuracy of each set of experiments.

There are several columns in the table for the SpiralFFT, each representing a maximum allowance for repeated angles. Since the length of the 1-D signal transformed with the CZT

Table 12: Algorithm performance as a function of the ratio of the spiral sampling density to the number of spirals. Each SpiralFFT column corresponds to a specific number of angle repetitions (“reps”), which allow for shorter 1-D signals and thus a more efficient transform. The average relative error rates are insensitive to the ratio of R/Q , and are given below each column. The term “relative error” refers to the ratio between the DFT values as computed by the SpiralFFT/NFFT3 and the direct calculation of the DFT value for a given frequency node. The error rates were calculated by averaging the error between the values returned by the algorithms and the “true” value calculated by explicitly evaluating (111) over 45 randomly chosen values of (r, q) . These 45 randomly chosen frequencies from each of the 9 rows were averaged, producing an average error rate over a total of 405 trials for each column of the table.

Ratio	SpiralFFT	SpiralFFT	SpiralFFT	SpiralFFT	SpiralFFT
R/Q	0 reps	1 reps	3 reps	7 reps	15 reps
1/8	7.99	6.47	5.83	5.24	4.87
1/6	8.83	7.74	6.68	5.98	5.39
1/4	11.41	9.98	7.92	7.42	6.20
1/2	16.71	14.70	11.13	9.77	8.68
1	26.93	22.22	17.26	14.15	13.20
2	44.21	34.83	27.26	21.37	19.37
4	76.46	57.56	43.32	34.69	29.37
6	103.54	79.15	56.79	46.16	37.70
8	135.10	98.92	73.26	61.64	47.15
Rel. Err.	4.99e-12	4.87e-12	5.00e-12	5.82e-12	5.24e-12
Ratio	NFFT3	NFFT3	NFFT3	NFFT3	NFFT3
R/Q	$(m = 2)$	$(m = 3)$	$(m = 4)$	$(m = 5)$	$(m = 6)$
1/8	15.62	27.38	42.30	63.15	91.86
1/6	15.30	26.21	40.63	61.06	89.17
1/4	14.41	24.24	38.26	57.94	85.50
1/2	13.26	22.13	35.23	53.59	79.11
1	11.70	19.76	32.11	49.68	74.00
2	10.54	18.24	29.85	47.00	70.44
4	9.58	16.80	28.36	45.31	68.34
6	9.19	16.26	27.51	44.23	67.01
8	9.12	15.80	27.29	44.09	66.22
Rel. Err.	4.87e-4	5.12e-6	5.44e-8	2.34e-9	5.82e-10

is dependent on the factors $L_1[r]$ and $L_2[r]$, it is desirable to reduce these values as much as possible. The more that angles are allowed to repeat (*i.e.*, use the same angles more than once as the spirals rotate about the alignment axis), the smaller the Farey sequence needed to generate the angles will be, and thus the smaller the average 1-D transform length will be. Allowing angles to be repeated does not affect the uniformity of the coverage of $[0, 2\pi)$, but it does reduce the “diversity” of the spiral contour. However, the effects of angle repetition on the conditioning of the frequency set are minimal (as is demonstrated in Section C.5.3), and as can be seen in Table 12, the difference in performance can be drastic.

There are also several columns in the table for the NFFT3, each corresponding to a different interpolation window cutoff. For each entry in the NFFT3 columns, the data sets from all of the corresponding row entries of the SpiralFFT portion of the table were processed, with the total execution time averaged. For example, the entry for $m = 2$, $R/Q = 1$ was generated by using the NFFT3 to calculate the Fourier coefficients for frequency sets with an R/Q ratio of 1 and angle repetitions of 0, 1, 3, 7, and 15, and then averaging the execution times across these 5 frequency sets.

As expected, the accuracy of the SpiralFFT does not vary as the angles are repeated. However, as the number of angle repetitions increases, the transform time drastically decreases. As the number of samples per spiral increases, the performance differences become more drastic. The transform speed decreases linearly (as predicted) as the ratio of R/Q increases.

Table 12 also reflects a distinct advantage in accuracy provided by the SpiralFFT. The NFFT3, even at a high cutoff parameter value, fails to achieve the same degree of accuracy as the SpiralFFT. This result is partially due to the second portion of the SpiralFFT being exact and the accuracy of the first portion only depending on the cutoff of an interpolation window for a 1-D NUFFT. Since the interpolation matrix can be precomputed, the extra cost introduced by using a slightly longer interpolation kernel for the 1-D NUFFTs is minimal.

C.5.2 Efficiency as a function of “oversampling ratio”

The effect of varying the so-called “oversampling ratio” – that is, the total number of Fourier coefficients computed relative to the spatial support of the signal – can also be studied with simulations. For this set of experiments, both the spatial support size and the number of k -space frequencies involved in the transform were varied to determine how the two algorithms compare to one another in various regimes. In all experiments, a fixed ratio R/Q of 2.5 was used. Further, since repeating the angles shows no significant problems in terms of the conditioning of the frequency set (as seen in Section C.5.3), the angles chosen from the Farey set were allowed to repeat up to 8 times within R samples. The NFFT3 was set to use an interpolation window cutoff of $m = 4$, which yields a relative error on the order of $1E-8$.

Figure 20 illustrates the algorithms’ performances over a range of spatial sizes and number of frequencies. In each figure, the y -axis represents the size of the spatial support (N^3) while the x -axis represents the number of k -space frequencies for which the transform is defined (QR). The plotted function is $e^{-0.01T_{ex}}$, where T_{ex} is the execution time for the algorithm. The exponential of the execution time is plotted to give better contrast in the image, and the value of 0.01 is chosen to yield an appropriate scale. Both images are scaled by the maximum execution time in the entire experiment for a fair comparison. Brighter values within the figure indicate faster execution times, darker values slower execution times.

The NUFFT follows the expected performance, growing linearly both with the quantity N^3 and the quantity QR (until memory considerations force a segmentation of the frequency set, at which point the growth in QR is no longer linear, as multiple FFTs must be computed to finished the transform). Similarly, the performance of the SpiralFFT algorithm follows the expected trends. For a fixed spatial size N^3 , as the value of QR increases, the execution time decreases. The same is true when N^3 is increased and QR is held constant. For the

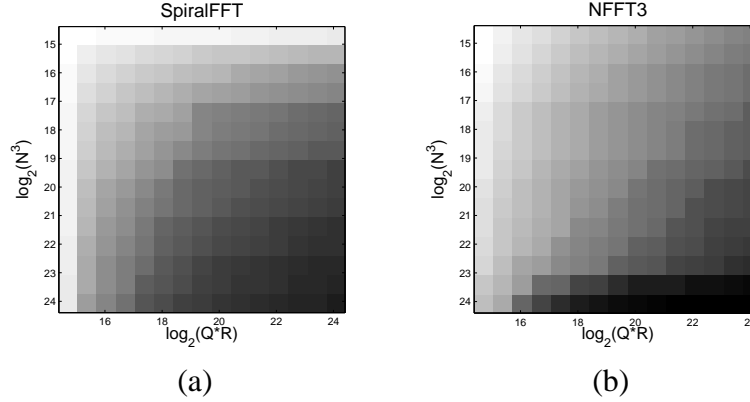


Figure 20: Graphical depiction of the performance of the (a) SpiralFFT and (b) NFFT3 over a range of spatial support sizes and number of frequency coefficients. Lighter values in the images indicate faster execution times. The performance of both algorithms increase as expected with increasing spatial support size and number of frequency coefficients. As the spatial support size grows very large, the NFFT3 must be broken up to deal with subsets of the frequency set, decreasing its computational efficiency.

portions of these figures corresponding to the typical problem size (columns 3-4, rows 5-6), the SpiralFFT is much more accurate (as shown in Section C.5.1) and executes slightly faster than the NUFFT.

The relative performance of the two algorithms is illustrated in Figure 21. The x -axis represents the total number of frequencies and the y -axis the size of the support (N) of a single spatial dimension, with the total amount of spatial data being N^3 . The curves plotted in Figure 21 depict contours along which the relative performance of the SpiralFFT compared to the NFFT3 is constant.

As seen in the figure, as the spatial dimension increases, the SpiralFFT becomes less efficient relative to the NFFT3. However, there is a fundamental limit to this behavior. The NFFT3 requires a full, oversampled 3-D FFT to be computed. At some point, the problem will be large enough that this cannot be accomplished within memory limits. However, the SpiralFFT only requires a maximum of $N^2 R_0$ complex numbers to be stored at once, where $R_0 \leq R$. This storage limit is in place because the frequencies can be subdivided in the variable R and the transform can be segmented in a natural way as it is performed. Therefore, there is some limit at which the NFFT3 will not be able to adequately store the amount

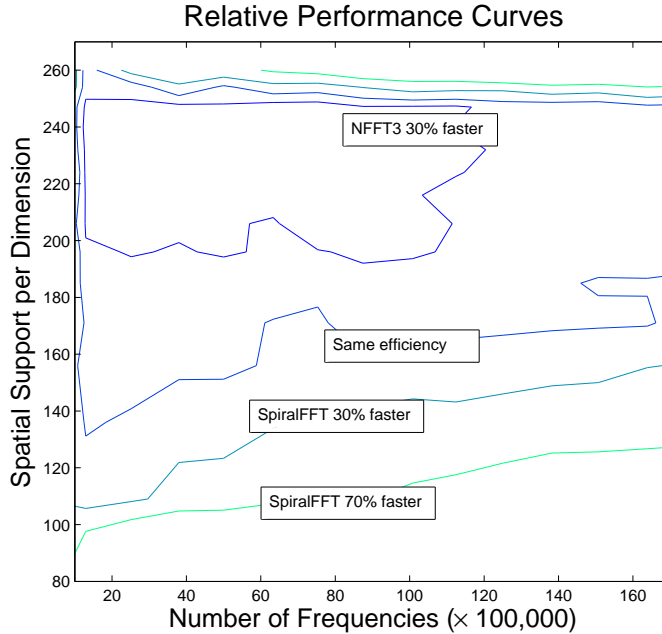


Figure 21: Relative performance curves for the NFFT3 and SpiralFFT. Each curve represents values of the spatial support N^3 and the number of frequencies QR for which the relative performance of the two algorithms remains fixed. In general, as the spatial support of the problem grows, the SpiralFFT becomes less efficient in a relative sense, while as the number of frequencies in the problem grows it becomes more efficient. After the spatial support becomes large enough, however, memory limitations cause the NUFFT to become less efficient in a relative sense, since the SpiralFFT does not require a full 3-D FFT to be computed.

of information needed to perform the transform, where the SpiralFFT may be broken into serial components to operate within memory constraints. In fact, this limit is not much further than the final row of this image; for $N^3 = 17.5M$, so much memory is taken up by the oversampled FFT portion of the NUFFT alone that a set of only $QR = 1.3M$ frequencies needed to be broken up and done sequentially. Therefore, for very resource-limited problems, the SpiralFFT poses an attractive alternative to the NUFFT. Further, Figure 21 shows that for a fixed spatial support, as the number of frequency coefficients increases, the relative performance of the SpiralFFT does as well.

C.5.3 Volume reconstructions

To verify that the spiral parametrizations proposed in the SpiralFFT algorithm are reasonable for use, LS reconstructions of several volumes from Fourier coefficients taken on

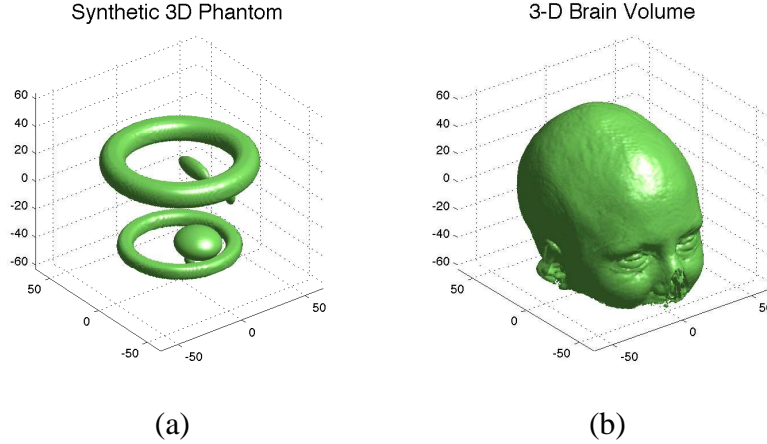


Figure 22: (a) Synthetic 3-D volume and (b) 3-D brain volume used for reconstruction experiments.

spirals matching the given parameterization requirements were computed. While an LS approach is not necessarily the modern technique of choice for reconstructing the volumes, it provides insight into whether a reasonable reconstruction is possible.

It is obvious that the functions $\alpha[r]$ and $\gamma[r]$ and the angles $\beta[r]$ could be chosen pathologically to yield unacceptable conditioning of the NUDFT matrix A . However, these parameters were chosen to mimic the behavior of spirals from spiral MRI data sets [51, 72]. The functions $\alpha[r]$ and $\gamma[r]$ were set to

$$\alpha[r] = \gamma[r] = \frac{1}{2} \sqrt{\frac{r}{R-1}} \quad r = 0, \dots, R-1.$$

These choices allow the spiral height and the spiral radius to grow smoothly without clustering too many samples about the k -space origin.

The angles $\beta[r]$ were allowed to be repeat between 0 and 15 times, with no noticeable change in the reconstruction artifacts. This result suggests that repetition of angles has little impact on the conditioning of the problem. In fact, the pSNR of the reconstructed volumes actually *increases* slightly as the number of repeated angles increases.

Synthetic 3-D volumes containing 3-D rings and oval shapes were reconstructed as well as a 3-D brain scan generated from freely available 2-D MR images. In all cases, the 3-D spatial support was a $64 \times 64 \times 64$ voxel cube, yielding a total of $0.26M$ spatial samples. For each reconstruction, $5.52M$ frequencies were generated with the prescribed sampling

patterns, with $Q = 1484$ and $R = 3720$. Six sets of spiral trajectories were sampled, with each set corresponding to an alignment axis and direction. Specifically, there were $Q_0 = Q_1 = Q_2 = Q_3 = 247$ spirals aligned with the positive and negative k_1 and k_2 axes, respectively, and $Q_4 = Q_5 = 248$ spirals aligned with the positive and negative k_3 axes.

Figure 22 illustrates the original synthetic and brain scan volumes. For the various angle repetitions, a single application of the SpiralFFT was used to generate the measurement data (*i.e.*, the desired Fourier coefficients). Once this data was generated, an approach similar to [36] was used to perform the volume reconstructions. Namely, the solution to

$$(A^HWA + \lambda I)x = A^Hb$$

was computed with CG, where F is the non-uniform Fourier matrix, W is a diagonal weighting matrix to compensate for the non-uniform sampling density in k -space, b is the vector of measurement data previously acquired, and λI is the Tikhonov regularization term. For the experiments, the Tikhonov parameter was set to $\lambda = 10^{-8}$. The values of W were chosen by calculating the volume of the 3-D Voronoi cells for the frequency set as described in [90]. Given the complexity of the frequency parametrization, analytic calculation of the weights is infeasible. Instead, the (finite) Voronoi diagram for the frequency set was computed, and the weight for each frequency point was calculated as the volume of the Voronoi cell centered at that point.

Since the matrix A^HWA has multi-level Toeplitz structure (see Section 3.1.5.5), instead of applying A^H and A separately each time the weighted Gramian A^HWA was to be applied the entire matrix was applied at once with 3-D FFTs. Each iteration of conjugate gradients executed in approximately 1 second, and the reconstructions were capped at a maximum iteration count of 1000.

Figure 23 shows slices of the least-squares reconstructions of the brain volume for no angle repetitions and seven angle repetitions, while Figure 24 shows similar content for the synthetic volume. As can be seen from the figures, the reconstructions are imperfect and spiral artifacts are present. However, these artifacts have no dependence on the number of

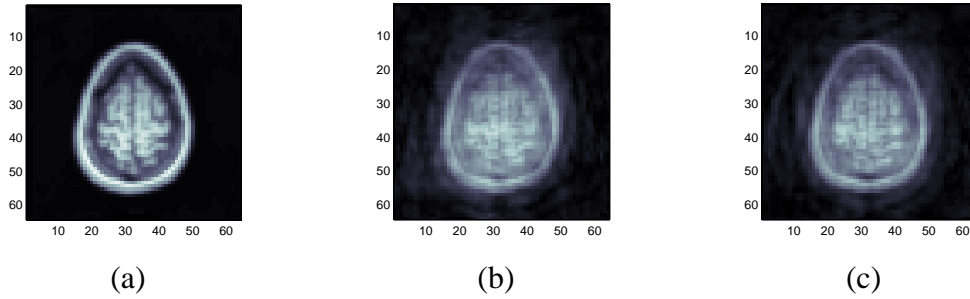


Figure 23: Brain volume slices for (a) input data, (b) least-squares reconstruction for a frequency set with angles repeated 0 times, (c) least-squares reconstruction for a frequency set with angles repeated seven times. Spiral artifacts are present in (b) and (c), but the magnitude of the artifacts does not increase as the number of angle repetitions increases.

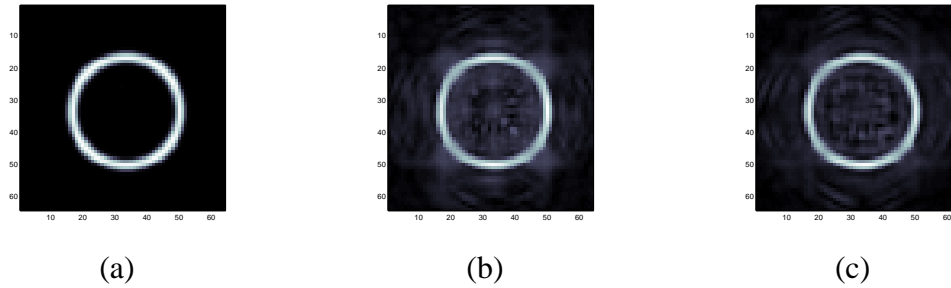


Figure 24: Synthetic phantom slices for (a) input data, (b) least-squares reconstruction for a frequency set with angles repeated 0 times, (c) least-squares reconstruction for a frequency set with angles repeated seven times. Spiral artifacts are present in (b) and (c), but the magnitude of the artifacts does not increase as the number of angle repetitions increases.

angle repetitions.

Table 13 compares the pSNR and mean-square error (MSE) for the reconstructed volumes as a function of the number of repeated angles. There is no appreciable difference in these quantities as the number of angle repetitions increases. The pSNR for the phantom image is significantly lower and the MSE is significantly higher than for the brain volume. These differences are due to the sparse nature of the synthetic phantom; unlike the brain volume, the vast majority of the synthetic phantom volume is empty. While perceptually the reconstruction quality is similar, this sparsity of data in the synthetic volume causes the pSNR and MSE of the reconstructions to be considerably worse than for the brain volume.

Table 13: *Peak signal-to-noise ratio and mean-square error for the least-squares reconstructed volumes. Both error metrics show little dependence on the number of repeated angles, with slight improvements as angles are repeated more frequently.*

Angle repetitions	pSNR	MSE
<i>Brain volume</i>		
1	21.69 dB	6.78E-3
2	21.84 dB	6.54E-3
4	22.47 dB	5.66E-3
8	23.23 dB	4.75E-3
16	23.97 dB	4.01E-3
<i>Synthetic phantom</i>		
1	1.95 dB	0.639
2	2.04 dB	0.625
4	2.45 dB	0.569
8	2.79 dB	0.529
16	2.77 dB	0.529

C.6 Summary

The SpiralFFT algorithm follows the examples of the Fast Slant Stack [6] (a fast Radon Transform) and the Polar FFT [5] (a fast Discrete Polar Fourier Transform) by using the structure of the sampling pattern to break expensive high-dimensional calculations into a larger number of cheaper, lower-dimensional calculations. For spiral sampling patterns fitting a specific set of conditions, the NUDFT summation of (111) can be calculated using a series of 1-D operations. Thus, the transform is effectively separable, a property which the general 3-D NUFFT does not share.

SpiralFFT was first outlined in [106], imposing a particularly restrictive sampling pattern. However, the algorithm was later generalized to loosen the requirements on the sampling modalities [107]. The broader parametrization of [107] removes the exactness of the transform, but imposes only a minor accuracy-*vs.*-efficiency trade-off. In addition, the new parametrization permits sampling patterns with better properties.

The experiments of Section C.5 demonstrate that the SpiralFFT can outperform a state-of-the-art NUFFT implementation in parameter regimes that are typical for 3-D spiral MRI problems. However, the potential gains are not universal across all problem sizes, and

there are parameter ranges for which the NUFFT is more efficient. The experiments also demonstrate that the performance of the SpiralFFT scales as expected with the various problem parameters.

REFERENCES

- [1] ADUKOV, V., “Generalized inversion of block Toeplitz matrices,” *Linear Algebra and Its Applications*, vol. 274, pp. 85–124, 1998. 4.2
- [2] AHN, C., KIM, J., and CHO, Z., “High-speed spiral-scan echo planar NMR imaging-I,” *IEEE Transaction on Medical Imaging*, vol. 5, no. 1, pp. 2–7, 1986. C.1.1
- [3] AMMAR, G. and GRAGG, W., “The generalized Schur algorithm for the superfast solution of Toeplitz systems,” in *Rational Approximation and Its Applications in Mathematics and Physics*, vol. 1237 of *Lecture Notes in Mathematics*, pp. 315–330, Springer, 1987. 1.3, 2.3.1
- [4] AMMAR, G. and GRAGG, W., “Superfast solution of real positive definite Toeplitz systems,” *SIAM Journal on Matrix Analysis and Applications*, vol. 9, pp. 61–76, 1988. 1.3, 2.3.1
- [5] AVERBUCH, A., COIFMAN, R., DONOHO, D., ELAD, M., and ISRAELI, M., “Accurate and fast Discrete Polar Fourier Transform,” *Conf. Record 37th Asilomar Conf. Sig., Sys., and Comp.*, vol. 2, pp. 1933–1937, 2003. C.6
- [6] AVERBUCH, A., COIFMAN, R., DONOHO, D., ISRAELI, M., and WALDEN, J., “Fast slant stack: a notion of Radon transform for data in a Cartesian grid which is rapidly computible, algebraically exact, geometrically faithful and invertible,” technical report, Statistics Department, Stanford University, 2001. C.6
- [7] BAREISS, E., “Numerical solution of linear equations with Toeplitz and vector Toeplitz matrices,” *Numerische Mathematik*, vol. 13, pp. 404–424, 1969. 1.3
- [8] BARNETT, S., “Polynomials and linear control systems,” in *Monographs and Textbooks in Pure and Applied Mathematics*, vol. 77, Marcel Dekker, Inc., 1983. 4.1.2.1, 1
- [9] BEN-ARTZI, A. and SHALOM, T., “On inversion of block Toeplitz matrices,” *Integral Equations and Operator Theory*, vol. 8, pp. 751–779, 1985. 4.2
- [10] BEN-ARTZI, A. and SHALOM, T., “On inversion of Toeplitz and close to Toeplitz matrices,” *Linear Algebra and Its Applications*, vol. 75, pp. 173–192, 1986. 4.3.3.1
- [11] BENEDETTO, J., *Wavelets: A tutorial in theory and applications*, ch. Irregular sampling and frames, pp. 445–507. Academic Press, New York, 1992. 3.2
- [12] BERNSTEIN, D., *Matrix mathematics: theory, facts, and formulas*. Princeton University Press, 2nd ed., 2009. A.2
- [13] BEYLKIN, G., “On the fast Fourier transform of functions with singularities.,” *Applied and Computational Harmonic Analysis*, vol. 2, no. 4, pp. 363–381, 1995. C.1

- [14] BINI, D., CODEVICO, G., and VAN BAREL, M., “Solving Toeplitz least squares problems by means of Newton’s iteration,” *Numerical Algorithms*, vol. 33, pp. 93–103, 2003. 3.1
- [15] BITMEAD, R. and ANDERSON, B., “Asymptotically fast solutions of Toeplitz and related systems of linear equations,” *Linear Algebra and Its Applications*, vol. 34, pp. 103–116, 1980. 1.3, 2.3.1, 2.3.1, 2
- [16] BRENT, R., GUSTAVSON, F., and YUN, D., “Fast solutions of Toeplitz systems of equations and computation of Padé approximations,” *Journal of Algorithms*, vol. 1, pp. 259–295, 1980. 1.2.2.3
- [17] BRUCKSTEIN, A. and KAILATH, T., “Inverse scattering for discrete transmission-line models,” *SIAM Review*, vol. 29, pp. 359–389, 1987. 1.2.2.3
- [18] BUTZER, P., SPLETTSTÖSSER, W., and STENS, R., “The sampling theorem and linear prediction in signal analysis,” *Jahresbericht der DMV*, vol. 90, pp. 1–70, 1988. 3.2
- [19] CARDINAL, J., “A divide-and-conquer method to solve Cauchy-like systems,” tech. rep., The FRISCO Consortium, 2000. 2.3.2
- [20] CARRIER, J., GREENGARD, L., and ROKHLIN, V., “A fast adaptive multipole algorithm for particle simulations,” *SIAM Journal on Scientific and Statistical Computing*, vol. 9, pp. 669–686, 1988. 1.3
- [21] CHANDRASEKARAN, S., DEWILDE, P., GU, M., PALS, T., SUN, X., VAN DER VEEN, A., and WHITE, D., “Some fast algorithms for sequentially semiseparable representations,” *SIAM Journal of Mathematical Analysis and Applications*, vol. 27, pp. 341–364, 2005. 1.3, 4.3.4.3
- [22] CHANDRASEKARAN, S., GU, M., SUN, X., XIA, J., and ZHU, J., “A superfast algorithm for Toeplitz systems of linear equations,” *SIAM Journal of Mathematical Analysis and Applications*, vol. 29, pp. 1247–1266, 2007. 1.3, 4.3.4.3
- [23] COLUCCIO, L., EISENBERG, A., and FEDELE, G., “A Prony-like method for non-uniform sampling,” *Signal Processing*, vol. 87, pp. 2484–2490, 2007. 3.2
- [24] COMMENGES, D. and MONSION, M., “Fast inversion of triangular Toeplitz matrices,” *IEEE Transactions on Automatic Control*, vol. AC-29, pp. 250–251, 1984. 4.1.2.2
- [25] COOLEY, J. and TUKEY, J., “An algorithm for the machine calculation of complex Fourier series,” *Mathematics of Computation*, vol. 19, pp. 297–301, 1965. 1.3
- [26] CORMEN, T., LEISERSON, C., RIVEST, R., and STEIN, C., *Introduction to Algorithms*. MIT Press and McGraw-Hill, second ed., 2001. 2.2.3.2
- [27] DE HOOG, F., “A new algorithm for solving Toeplitz systems of equations,” *Linear Algebra and Its Applications*, vol. 88/89, pp. 123–138, 1987. 1.3

- [28] DURBIN, J., “The fitting of time series models,” *Review of the International Statistical Institute*, vol. 28, pp. 233–243, 1960. 1.2.2.3, 1.3
- [29] DUTT, A. and ROKHLIN, V., “Fast Fourier transforms for nonequispaced data,” *SIAM Journal of Scientific Computing*, vol. 14, no. 6, pp. 1368–1393, 1993. C.1, C.1.2
- [30] DUTT, A. and ROKHLIN, V., “Fast Fourier transforms for nonequispaced data, ii.,” *Applied Computational Harmonic Analysis*, vol. 2, pp. 85–10, 1995. 1.3
- [31] ELLIS, R. and GOHBERG, I., “Inversion formulas for infinite generalized Toeplitz matrices,” *Integral Equation and Operator Theory*, vol. 32, pp. 29–64, 1998. 4.2
- [32] ELLIS, R., GOHBERG, I., and LAY, D., “Infinite analogues of block Toeplitz matrices and related orthogonal functions,” *Integral Equations and Operator Theory*, vol. 22, pp. 375–419, 1995. 4.2
- [33] ENG, F., *Non-uniform sampling in statistical signal processing*. PhD thesis, Linköping University, 2007. 3.2
- [34] FARENICK, D., KRUPNIK, M., KRUPNIK, N., and LEE, W., “Normal Toeplitz matrices,” *SIAM Journal on Matrix Analysis and Applications*, vol. 17, pp. 1037–1043, 1996. 4.1.1
- [35] FEICHTINGER, H. and GRÖCHENIG, K., *Wavelets: Mathematics and Applications*, ch. Theory and practice of irregular sampling, pp. 305–363. CRC Press, Boca Raton, FL, 1994. 3.2
- [36] FEICHTINGER, H., GRÖCHENIG, K., and STROHMER, T., “Efficient numerical methods in non-uniform sampling theory,” *Numerische Mathematik*, vol. 69, pp. 423–440, 1995. 3.2, C.5.3
- [37] FESSLER, J. A. and SUTTON, B. P., “Nonuniform fast Fourier transforms using min-max interpolation,” *IEEE Transaction on Signal Processing*, vol. 51, no. 2, pp. 560–574, 2003. C.1.2
- [38] FESSLER, J., LEE, S., OLAFSSON, V., SHI, H., and NOLL, D., “Toeplitz-based iterative image reconstruction for MRI with correction for magnetic field inhomogeneity,” *IEEE Transactions on Signal Processing*, vol. 32, 2005. 3.1.5
- [39] FIEDLER, M., “Hankel and Loewner matrices,” *Linear Algebra and Its Applications*, vol. 58, pp. 75–95, 1984. 1.3
- [40] FRIEDLANDER, B., MORE, M., KAILATH, T., and LJUNG, L., “New inversion formulas for matrices classified in terms of their distances from Toeplitz matrices,” *Linear Algebra and Its Applications*, vol. 27, pp. 31–60, 1979. 1.4.1
- [41] FRIGO, M., *FFTW3 Library*. Massachusetts Institute of Technology. Software Library. C.4, C.5

- [42] FUHRMANN, P., *A polynomial approach to linear algebra*. Springer-Verlag, New York, Inc., 1996. 4.1.1
- [43] GOHBERG, I. and HEINIG, G., “Inversion of finite Toeplitz matrices composed of elements of a noncommutative algebra,” *Revue Roumaine de Mathématiques Pures et Appliquées*, vol. 19, pp. 623–663, 1974. 4.2
- [44] GOHBERG, I., KAILATH, T., and OLSHEVSKY, V., “Fast Gaussian elimination with partial pivoting for matrices with displacement structure,” *Mathematics of Computation*, vol. 64, pp. 39–59, 1995. 1.3, 2.3.2
- [45] GOHBERG, I. and KRUPNIK, N., “A formula for the inversion of finite Toeplitz matrices (in Russian),” *Matematicheskie Issledovaniya*, vol. 7, pp. 272–283, 1972. 4.3.3.1
- [46] GOHBERG, I., LANCASTER, P., and RODMAN, L., *Invariant subspaces of matrices with applications*. John Wiley & Sons, New York, 1986. 4.1.1
- [47] GOHBERG, I. and OLSHEVSKY, V., “Complexity of multiplication with vectors for structured matrices,” *Linear Algebra and Its Applications*, vol. 202, pp. 163–192, 1994. 1.2.2.3, 1.3
- [48] GOHBERG, I. and SEMENCUL, A., “On the inversion of finite-section Toeplitz matrices and their continuous analogues (in Russian),” *Matematicheskie Issledovaniya*, vol. 7, pp. 201–224, 1972. 1.3, 2
- [49] GREENGARD, L. and LEE, J., “Accelerating the nonuniform fast Fourier transform,” *SIAM Review*, vol. 46, no. 3, pp. 443–454, 2004. C.1.2
- [50] GREENGARD, L. and ROKHLIN, V., “A fast algorithm for particle simulations,” *Journal of Computational Physics*, vol. 73, pp. 325–348, 1987. 1.2.1.3
- [51] GURNEY, P., HARGREAVES, B., and NISHIMURA, D., “Design and analysis of a practical 3d cones trajectory,” *Magnetic Resonance in Medicine*, vol. 55, pp. 575–82, 2006. C.5.1, C.5.3
- [52] HEINIG, G., *Inversion of generalized Cauchy matrices and other classes of structured matrices*, vol. 69 of *Linear Algebra for Signal Processing*, pp. 63–81. Springer, 1995. 1.3, 2.2.2.1, 2.2.2.4
- [53] HEINIG, G., “Solving Toeplitz systems after transformation and extension,” *Calcolo*, vol. 33, pp. 115–129, 1996. 2.2.3.2, 3.1.4.2, 5.1, A.4
- [54] HEINIG, G., “On the reconstruction of Toeplitz matrix inverses from columns,” *Linear Algebra and Its Applications*, vol. 350, pp. 199–212, 2002. 4.3.3, 4.3.3.1
- [55] HEINIG, G., “Fast algorithms for Toeplitz least squares problems,” *Operator Theory: Advances and Applications*, vol. 149, pp. 167–197, 2004. 3.1
- [56] HEINIG, G. and ROST, K., *Algebraic Methods for Toeplitz-like Matrices and Operators*. Akademie-Verlag, Berlin, and Birkhäuser, Boston, 1984. 1.3, 1

- [57] HUCKLE, T., “Computation of Gohberg-Semencul formulas for a Toeplitz matrix,” Tech. Rep. SSCM-93-14, Computer Science Department, Stanford University, Stanford, CA, November 1993. 3.1.5.3
- [58] JACKSON, J. J., MEYER, C. H., NISHIMURA, D. G., and MACOVSKI, A., “Selection of a convolution function for Fourier inversion using gridding,” *IEEE Transaction on Medical Imaging*, vol. 10, no. 3, pp. 473–478, 1991. C.1.2
- [59] JEANNEROD, C. and MOUILLERON, C., “Computed specified generators of structured matrix inverses,” *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pp. 281–288, 2010. 3.1.3
- [60] KAILATH, T., “Some new algorithms for recursive estimation in constant linear systems,” *IEEE Transactions on Information Theory*, vol. 19, pp. 750–760, 1973. 1.4.1
- [61] KAILATH, T., “Remarks on the origin of the displacement-rank concept,” *Applied Mathematics and Computation*, vol. 45, pp. 193–206, 1991. 1.4.1
- [62] KAILATH, T. and KOLTRACHT, I., “Matrices with block Toeplitz inverses,” *Linear Algebra and Its Applications*, vol. 75, pp. 145–153, 1986. 4.2
- [63] KAILATH, T., KUNG, S., and MORF, M., “Displacement ranks of matrices and linear equations,” *Journal of Mathematical Analysis and Applications*, vol. 68, pp. 395–407, 1979. 1.3, 1.4.1, 3, 4.3.2
- [64] KEINER, J., KUNIS, S., and POTTS, D., *NFFT3 Library*. Faculty of Mathematics of the Chemnitz University of Technology and at the Mathematical Institute of the University of Lübeck. Software Library. C.5
- [65] KEINER, J., KUNIS, S., and POTTS, D., “Using NFFT3: a software library for various nonequispaced Fast Fourier Transforms,” *ACM Transactions on Mathematical Software*, vol. 36, pp. 19:1–19:30, 2009. 3.2.5.1, C.1, C.2.2, C.4
- [66] KRAVANJA, P. and BAREL, M., “Coupled Vandermonde matrices and the superfast computation of Toeplitz determinants,” *Numerical Algorithms*, vol. 24, pp. 99–116, 2000. 2.2.1
- [67] LABAHN, G., CHOI, D., and CABAY, S., “The inverses of block Hankel and block Toeplitz matrices,” *SIAM Journal on Computing*, vol. 19, pp. 98–123, 1990. 4.2
- [68] LABAHN, G. and SHALOM, T., “Inversion of Toeplitz matrices with only two standard equations,” *Linear Algebra and Its Applications*, vol. 175, pp. 143–158, 1992. 4.3.3.1
- [69] LERER, L. and TISMENETSKY, M., “Generalized Bezoutian and the inversion problem for block matrices, I. general scheme,” *Integral Equations and Operator Theory*, vol. 9, pp. 790–819, 1986. 4.2

- [70] LEVINSON, N., “The Wiener RMS error criterion in filter design and prediction,” *Journal of Mathematical Physics*, vol. 25, pp. 261–278, 1947. 1.3
- [71] LV, X. and HUANG, T., “The inverses of block Toeplitz matrices,” *Journal of Mathematics*, vol. 2013, pp. 1–8, 2013. 4.2
- [72] MAKHIJANI, M. and NAYAK, K., “3d cones with trajectory with anisotropic field-of-view,” *Proc. ISMRM 17th Scientific Sessions, Honolulu*, p. 2622, 2009. C.5.1, C.5.3
- [73] MARTINSSON, P., ROKHLIN, V., and TYGERT, M., “A fast algorithm for the inversion of general Toeplitz matrices,” *Computers and Mathematics with Applications*, vol. 50, pp. 741–752, 2005. 1.3
- [74] MARVASTI, F., *Advanced Topics in Shannon Sampling and Interpolation Theory*, ch. Nonuniform sampling, pp. 121–156. Springer, Berlin, 1993. 3.2
- [75] MEYER, C. H., HU, B. S., NISHIMURA, D. G., and MACOVSKI, A., “Fast spiral coronary artery imaging,” *Magnetic Resonance in Medicine*, vol. 28, pp. 202–213, 1992. C.1.1
- [76] MORF, M., *Fast algorithms for multivariable systems*. PhD thesis, Stanford University, 1974. 1.3
- [77] MORF, M., “Doubling algorithms for Toeplitz and related equation,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1980. 1.3, 2.3.1
- [78] MOUILLERON, C., *Efficient computation with structured matrices and arithmetic expressions*. PhD thesis, École Normale Supérieure de Lyon, 2011. 2.3.2
- [79] MOURRAIN, B. and PAN, V., “Multivariate polynomials, duality, and structured matrices,” *Journal of Complexity*, vol. 16, no. 1, pp. 110–180, 2000. 4.3.2, 1
- [80] MUSICUS, B., “Levinson and fast Choleski algorithms for Toeplitz and almost Toeplitz matrices,” tech. rep., Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1984. 1.3
- [81] O’SULLIVAN, J., “A fast sinc function gridding algorithm for Fourier inversion in computer tomography,” *IEEE Transactions on Medical Imaging*, vol. 4, no. 4, pp. 200–207, 1985. C.1.2
- [82] PAN, V., “On computations with dense structured matrices,” *Mathematics of Computation*, vol. 55, pp. 179–190, 1990. 1.3, 1
- [83] PAN, V., *Structured matrices and polynomials: unified superfast algorithms*. Birkhauser/Springer, 2001. 1.2.1.3, 1.2.1.3, 1.2.1.4, 1.4.1, 1, 2, 1.4.4, 2.3.1, 4.2.2
- [84] PAN, V., VAN BAREL, M., WANG, X., and CODEVICO, G., “Iterative inversion of structured matrices,” *Theoretical Computer Science*, vol. 315, pp. 581–592, 2004. 3.1

- [85] PAN, V. and WANG, X., “Inversion of displacement operators,” *SIAM Journal on Matrix Analysis and Applications*, vol. 24, pp. 660–677, 2003. 1.4.3, 1.4.3
- [86] POLONI, F., “A note on the $o(n)$ -storage implementation of the GKO algorithm and its adaption to trummer-like matrices,” *Numerical Algorithms*, pp. 115–139, 2010. 2.3.2
- [87] POTTS, D. and STEIDL, G., “Fast summation at nonequispaced knots by NFFTs,” *SIAM Journal on Scientific Computing*, vol. 24, pp. 2013–2037, 2003. 3.2.5.1
- [88] POTTS, D., STEIDL, G., and TASCHE, M., “Fast Fourier transforms for nonequispaced data: a tutorial,” in *Modern Sampling Theory: Mathematics and Applications* (BENEDETTO, J. and FERREIRA, P., eds.), ch. 12, pp. 249–274, Birkhäuser, 2001. C.1.2
- [89] RABINER, L., SCHAFTER, R., and RADER, C., “The chirp z-Transform algorithm,” *IEEE Transactions on Audio an Electroacoustics*, vol. 17, no. 2, pp. 86–92, 1969. C.2.2
- [90] RASCHE, V., PROKSA, R., SINKUS, R., BÖRNERT, P., and EGGERS, H., “Resampling of data between arbitrary grids using convolution interpolation,” *IEEE Transactions on Medical Imaging*, vol. 18, pp. 385–392, 1999. 3.1.5.5, C.5.3
- [91] RUSSAKOVSKI, E., *On Hankel and Toeplitz matrices and the Bezoutian II.*, vol. 33 of *Theory of Functions.*, pp. 119–124. Functional Analysis and Applications, 1980. 1
- [92] SANKAR, P. and FERRARI, L., “Simple algorithms and architectures for B-spline interpolation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 271–276, 1988. 1.2.2.3
- [93] SARTY, G. E., BENNETT, R., and COX, R. W., “Direct reconstruction of non-Cartesian k-space data using a nonuniform fast Fourier transform,” *Magnetic Resonance in Medicine*, vol. 45, pp. 908–915, 2001. C.1.2
- [94] SCHUR, I., “Über Potenzreihen, die im innern des einheitskreises beschränkt sind,” *Journal für die Reine und Angewandte Mathematik*, vol. 147, pp. 205–232, 1917. 3
- [95] SEXTON, H., “An analysis of an algorithm of Bitmead and Anderson for the inversion of Toeplitz systems,” Tech. Rep. 756, Naval Oceans System Center, 1982. 1.3
- [96] SHA, L., GUO, H., and SONG, A. W., “An improved gridding method for spiral MRI using nonuniform fast Fourier transform,” *Journal of Magnetic Resonance*, vol. 162, pp. 250–258, 2003. C.1.2
- [97] SIDDIQUI, M., “On the inversion of the sample covariance matrix in a stationary autoregressive process,” *The Annals of Mathematical Statistics*, vol. 19, pp. 585–588, 1958. 1.3
- [98] STEIDL, G., “A note on fast Fourier transforms for nonequispaced grids,” *Advances in Computational Mathematics*, vol. 9, pp. 337–353, 1998. C.1

- [99] STEWART, M., “A superfast Toeplitz solver with improved numerical stability,” *SIAM Journal on Matrix Analysis and Applications*, vol. 25, pp. 669–693, 2003. 1.3
- [100] STROHMER, T., “Numerical analysis of the non-uniform sampling problem,” *Journal of Computational and Applied Mathematics*, vol. 122, pp. 297–316, 2000. 3.2
- [101] SWEET, D., “Fast Toeplitz orthogonalization,” *Numerische Mathematik*, vol. 43, pp. 1–21, 1984. 3.1
- [102] TRENCH, W., “An algorithm for the inversion of finite Toeplitz matrices,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 12, pp. 512–522, 1964. 1.3
- [103] TURNES, C., BALCAN, D., and ROMBERG, J., “Image deconvolution via superfast inversion of a class of two-level Toeplitz matrices,” *Proceedings of the IEEE International Conference on Image Processing*, pp. 3073–3076, 2012. 4.1
- [104] TURNES, C., BALCAN, D., and ROMBERG, J., “Superfast Tikhonov regularization of Toeplitz systems,” *IEEE Transactions on Signal Processing*, Accepted for publication (February 2014). 3.1
- [105] TURNES, C., BALCAN, D., and ROMBERG, J., “One-level formulas for the inverses of two-level Toeplitz matrices,” *SIAM Journal on Matrix Analysis and Applications*, In preparation. 4.2
- [106] TURNES, C. and ROMBERG, J., “SpiralFFT: An efficient method for 3-D FFTs on spiral MRI contours,” *Proceedings of the IEEE International Conference on Image Processing*, pp. 617–620, 2010. C, C.2.3, C.6
- [107] TURNES, C. and ROMBERG, J., “Efficient calculations of 3-D FFTs on spiral contours,” *Journal of Scientific Computing*, vol. 50, pp. 610–628, 2012. C, C.6
- [108] VAN BAREL, M. and BULTHEEL, A., “A general module theoretic framework for vector M-Padé and matrix rational interpolation,” *Numerical Algorithms*, vol. 3, pp. 451–461, 1992. 1.3, 2.2.3, 2.2.3.1, 2, 2.2.3.1, 4.3.4.3
- [109] VAN BAREL, M. and BULTHEEL, A., *The “look-ahead” philosophy applied to matrix rational interpolation problems*, vol. Systems and Networks: Mathematical Theory and Applications, Vol. II: Invited and Contributed Papers of *Mathematical Research* 79, pp. 891–894. Akademie-Verlag, Berlin, 1994. 2.2.3.2
- [110] VAN BAREL, M., HEINIG, G., and KRAVANJA, P., “A stabilized superfast solver for non-symmetric Toeplitz systems,” *SIAM Journal of Matrix Analysis and Applications*, vol. 23, pp. 494–510, 2001. 1.3, 2.2.2.2, 2.2.2.3, 2, 3.1.4.2, 3.1.5.2
- [111] VAN BAREL, M., HEINIG, G., and KRAVANJA, P., “A superfast method for solving Toeplitz linear least squares problems,” *Linear Algebra and Its Applications*, vol. 366, pp. 441–457, 2003. 2.2.2.3, 2, 3.1, 3.1.1, 3.1.4.1, 3.1.4.2, 4.1.3

- [112] VAN BAREL, M. and KRAVANJA, P., “A stabilized superfast solver for indefinite Hankel systems,” *Linear Algebra and Its Applications*, vol. 284, pp. 335–355, 1998. 2.2.3.2
- [113] WARE, A., “Fast approximate Fourier transforms for irregularly spaced data,” *SIAM Review*, vol. 40, pp. 838–856, 1998. C.1
- [114] WINGHAM, D., “The reconstruction of a band-limited function and its fourier transform from a finite number of samples at arbitrary locations by singular value decomposition,” *IEEE Transaction on Circuit Theory*, vol. 40, pp. 559–570, 1992. 3.2
- [115] WISE, J., “The autocorrelation function and the spectral density function,” *Biometrika*, vol. 42, pp. 151–159, 1955. 1.3
- [116] ZOHAR, S., “Toeplitz matrix inversion: the algorithm of W. F. Trench,” *Journal of the Association for Computing Machinery*, vol. 16, pp. 592–601, Oct. 1969. 1.3