4-21-2008

Algorithms for Toeplitz Matrices with Applications to Image Deblurring

Symon Kipyagwai Kimitei

ALGORITHMS FOR TOEPLITZ MATRICES WITH APPLICATIONS TO IMAGE DEBLURRING

by

Symon Kimitei

Under the Direction of Michael Stewart

ABSTRACT

In this thesis, we present the $O(n \log^2 n)$ superfast linear least squares Schur algorithm (sflssschur). The algorithm we will describe illustrates a fast way of solving linear equations or linear least squares problems with low displacement rank. This program is based on the $O(n^2)$ Schur algorithm speeded up via FFT. The algorithm solves a ill-conditioned Toeplitz-like system using Tikhonov regularization. The regularized system is Toeplitz-like of displacement rank 4. We show the effect of choice of the regularization parameter on the quality of the image reconstruction.

INDEX WORDS: Toeplitz matrix, Schur algorithm, Ill-posed problem, Tikhonov Regularization, Gohberg-Semencul formula.

ALGORITHMS FOR TOEPLITZ MATRICES WITH APPLICATIONS TO IMAGE DEBLURRING

by

Symon Kimitei

A Thesis Submitted in Partial Fulfillment of Requirements for the Degree of

Master of Science

in the College of Arts and Sciences

Georgia State University

2008

ALGORITHMS FOR TOEPLITZ MATRICES WITH APPLICATIONS TO IMAGE DEBLURRING

by

Symon Kimitei

Major Professor:	Michael Stewart
Committee:	George Davis
	Valerie Miller

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
March 2008

Contents

List of Figures

List of Tables

1 Introduction

In this research effort, the basic problem consists of the restoration of images which have been degraded by noise. The idea is that given a degraded image, it is possible to use mathematical procedures to improve the quality of the image. There are several areas of digital image processing of interest to scientists and these include: image restoration, image enhancement, image compression and image recognition. We will dwell entirely in the area of image restoration using Toeplitz systems. This field of image restoration is an important part of digital image processing and plays a key role in many applications today such as in satellite, military reconnaissance missions, medical, forensic science and astronomical imaging. It is also applicable in the restoration of poor-quality family portraits.

When we talk of a degraded image, we imply that the image has noise in it. The noise in our images may arise naturally as in poorly stored family pictures where the degradation may be caused by environmental conditions. However, there are other cases in which the noise may be induced by the type of equipment used to capture the images. In any case, our quest in this thesis is to denoise the given image by solving an ill-conditioned linear least squares problem. A more explicit explanation of image noise and how it arises can be found in [9] and [10].

In digital image processing, an image is represented by a $2-\mathrm{D}$ signal otherwise called an array of matrix coefficients. Before we describe the blurred image representation, it is important that we define Toeplitz matrices. This type is of greater importance to us because its structure is worthy of exploitation. A Toeplitz matrix is defined as

$$T = \begin{pmatrix} s_0 & s_1 & \cdots & s_{n-1} \\ s_{-1} & s_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ s_{-(m-1)} & s_{-(m-2)} & \cdots & s_{-(m-n+1)} \end{pmatrix} \tag{1}$$

a matrix with constants along the diagonals from top left to bottom right. In a more general

case, we define $s_{ij} = s_{j-i}$ such that $0 \leq i \leq n-1$ and $0 \leq j \leq m-1$. All the matrix coefficients are scalar quantities. When a matrix is filled with coefficients which exhibit a structure such as T above, we refer such a matrix as Toeplitz. The values of s_{j-i} are defined using the point spread function(p.s.f), $\phi(x,y) = g(x)\,h(y)$. In the algorithms used in this thesis project, we will choose our p.s.f to be

$$g(x) = e^{-0.1\left(x^2\right)}$$

and

$$h(y) = e^{-0.1\left(y^2\right)}$$

because they have zero boundary conditions. Since the p.s.f is seperable, it implies that our blurring model $B_{ij} = B_{xy}$. The values $0 \leq x \leq n-1$ and $0 \leq y \leq m-1$. Given the definitions, it is easy to see that the blurring model can be constructed as

$$\begin{aligned} B_{xy} &= \sum_{\hat{x}\hat{y}} I(\hat{x},\hat{y})\,\phi(x-\hat{x},y-\hat{y}) \\ &= \sum_{\hat{x}\hat{y}} I(\hat{x},\hat{y})\,g(x-\hat{x})\,h(y-\hat{y}) \\ &= \sum_{\hat{x}} g(x-\hat{x}) \left(\sum_{\hat{y}} I(\hat{x},\hat{y})\,h(y-\hat{y}) \right) \end{aligned} \tag{2}$$

If we let $I_1(x,y) = \sum_{\hat{y}} I(x,\hat{y})\,h(y-\hat{y})$ then

$$I_1 = \begin{pmatrix} i_{11} & i_{12} & \cdots & i_{1n} \\ \vdots & \vdots & \cdots & \vdots \\ i_{(n-1)1} & i_{(n-1)2} & \cdots & i_{(n-1)(n-1)} \\ i_{n1} & i_{n2} & \cdots & i_{nn} \end{pmatrix} \begin{pmatrix} h_0 & h_1 & \cdots & h_{n-1} \\ h_{-1} & h_0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & h_1 \\ h_{-(n-1)} & h_{-(n-2)} & \cdots & h_0 \end{pmatrix}.$$

This indicates that $I_1(x,y) = I(\hat{x},\hat{y})\,T_1$. Consequently, we can express the blurred model

$B = I_2$ such that

$$
\begin{aligned}
I_2 &= \sum_{\hat{x}} g\left(x - \hat{x}\right) I_1\left(\hat{x}, y\right) \\
&= \begin{pmatrix} g_0 & g_1 & \cdots & g_{n-1} \\ g_{-1} & g_0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & g_1 \\ g_{-(n-1)} & g_{-(n-2)} & \cdots & g_0 \end{pmatrix} \begin{pmatrix} i_{11}^{(1)} & i_{12}^{(1)} & \cdots & i_{1n}^{(1)} \\ \vdots & \vdots & \cdots & \vdots \\ i_{(n-1)1}^{(1)} & i_{(n-1)2}^{(1)} & \cdots & i_{(n-1)(n-1)}^{(1)} \\ i_{n1}^{(1)} & i_{n2}^{(1)} & \cdots & i_{nn}^{(1)} \end{pmatrix} \\
&= T_2 \quad I\left(\hat{x}, \hat{y}\right) T_1
\end{aligned}
$$

The matrices T_1 and T_2 represent matrix convolution by the point spread functions $g(x)$ and $h(y)$. As seen in the blurred model representation (2), the p.s.f used is seperable which means that we can express it as a product of two convolution matrices, T_1 and T_2 .

In particular, if given an image represented by $X \in \Re^{n \times n}$, we will assume that X is blurred and corrupted by randomized noise, $F \in \Re^{n \times n}$. The random noise is small relative to the blurred image $B \in \Re^{n \times n}$. Therefore, the blurred image actually seen is given by the equation

$$
\begin{aligned}
B &= T_1 \hat{X} T_2 + F \\
\hat{X} &= T_1^{-1} B T_2^{-1} - T_1^{-1} F T_2^{-1}
\end{aligned}
$$

Our discussion of the blurred model involves substituting $\hat{X} = I$ used in the explanation of the seperable p.s.f to avoid confusion with the standard identity matrix, I.

Therefore, in the blurred model representation, both T_1 and T_2 are square matrices and the matrix $B \in \Re^{n \times n}$ is the blurry image which is represented by the original sharp image X, the induced noise $F \in \Re^{n \times n}$, $T_1 \in \Re^{n \times n}$ and $T_2 \in \Re^{n \times n}$ both of which are square Toeplitz matrices. The noise matrix F occurs in the form of image degrading features as described earlier. So in an ideal sense, when an image is not degraded, we can define the blurred image

matrix as

$$B = T_1 X T_2.$$

As noted above, if T_1 or T_2 is ill-conditioned, it may not be sufficient to compute

$$X = T_1^{-1} B T_2^{-1}.$$

This implies that if we proceed to compute the image error, we obtain

$$\begin{aligned} X - \hat{X} &= T_1^{-1} B T_2^{-1} - \left(T_1^{-1} B T_2^{-1} - T_1^{-1} F T_2^{-1} \right) \\ &= T_1^{-1} F T_2^{-1} \end{aligned} \tag{3}$$

The result (3) indicates that the error in the system becomes magnified when the problem we solve is ill-conditioned. If $\|T_1^{-1}\|_2$ or $\|T_2^{-1}\|_2$ is large, then the error in the solution is large as well. If the noise present is also larger, the error becomes even more magnified. To rectify the problem of ill-conditioning, we apply regularization which allows for the stabilization of our solutions. There are many other ways to regularize a system. The mathematical operators often employed include filtering regularization techniques such as: Pseudoinverse, Wiener, Tikhonov and more. Others may be iterative such as: Richardson-Lucy, total variation, conjugate gradient and more .

In classic image restoration techniques, the image produced is simply the sum of noise and the blurring operator acting on the true image as indicated above. In [5], the blurring operator described is regarded as a convolution with a point-spread function that characterizes a deviation from the true image. In digital image processing, the techniques often employed in image deblurring seek to recover the degraded image by solving an ill-conditioned least squares problem, usually regularized through the addition of some smoothness requirement to be satisfied by the restored image.

Clearly, the linear problem being solved is a formidable computation because of the size of the blurring operator whose dimensions are the number of pixels in the image and the

potential problem of ill conditioning. In [5], it is suggested that if we apply regularization to such ill-conditioned problems, one can render them practically solvable. However, the article alludes to the fact that regularization can remove sharp edges and similar image distinguishing features during the process. This means that when we apply the regularization technique, our quest is simply to try to stabilize the ill-conditioned problem and solve it without removing image distinguishing features. When this is done, it is possible to attain a satisfactory solution to the problem.

In our research, we employ the Tikhonov regularization technique to solve the linear least squares problem. A linear least squares method finds the best-fit or the best approximation to the linear problem. The goal then is to minimize the norm of the residual. As indicated earlier, this method allows us to find an approximate solution to the image restoration problem by practically neutralizing the ill-conditioning associated with it. To do so, we will use a regularization parameter which is a controlled scalar quantity defined as α. The parameter is varied to determine an optimal level of the minimized residual norm. When we strike a reasonable balance between the non-zero residual and the solution norm, the solution attained is referred to as the Tikhonov solution [2]. In effect, the solution of the linear least squares system obtained is an approximation of the actual solution.

At the heart of the problem being solved is the application of the fast generalized Schur algorithm which is known to factorize a symmetric, positive definite Toeplitz-like matrix within a runtime of $\mathrm{O}(n^2)$. This is in contrast to the slow $\mathrm{O}(n^3)$ algorithms often used to solve linear systems of equations for example the Gaussian elimination procedure. Solving linear equations sometimes involves decomposing the matrix being solved into upper and lower triangular and this will be the case in our algorithms. It is therefore worth noting that the generalized Schur algorithm is a proper choice because it decomposes any symmetric positive definite Toeplitz-like matrix via the Cholesky factorization process to yield C^TC, where C is the upper triangular Cholesky factor.

In order to speed up the computations of the generalized Schur algorithm, the superfast

linear least squares Schur algorithm is used. The basis of this algorithm is the generalized Schur algorithm but speeded up computationally via FFT. The idea behind the linear least squares problem is that one can find the minimization to the residual

$$\min_{Y} \| \begin{pmatrix} T \\ \alpha I \end{pmatrix} Y - \begin{pmatrix} B \\ 0 \end{pmatrix} \|_F^2$$

using the Tikhonov regularization where α is called the regularization parameter and I is the identity matrix. Using this concept, it means that it is possible for one to solve the normal equations

$$\begin{aligned} \begin{pmatrix} T^T & \alpha I \end{pmatrix} \begin{pmatrix} T \\ \alpha I \end{pmatrix} Y &= \begin{pmatrix} T^T & \alpha I \end{pmatrix} \begin{pmatrix} B \\ 0 \end{pmatrix} \\ \left(T^T T + \alpha^2 I\right) Y &= T^T B \end{aligned}$$

Since

$$\left(T^T T + \alpha^2 I\right) = C^T C$$

it means that by the Cholesky factorization procedure, the normal equations can then be reformulated so that the linear equation we will use to solve for Y becomes

$$\left(C^T C\right) Y = T^T B$$

Note that this equation can be solved easily by back substitution since C is an upper triangular matrix. This problem has a runtime of $O\left(n^3\right)$.

We start the thesis project with the subsection 1.1 examining the problem of ill-conditioning and show that we are indeed solving a perturbed system. Here, we explain what a system condition number is and show its relevance to our project. In subsection 1.2, we discuss the Tikhonov regularization procedure and introduce the concept of a regularization parameter which is used in solving the least squares problem. This is immediately followed by

an overview of hyperbolic and plane rotations in subsections 1.3 and 1.4. Hyperbolic and plane rotations are fundamental tools in the Schur algorithm. In these two subsections, we will discuss what hyperbolic and plane rotations are and explain how they can be used to introduce zeros in a matrix or a vector. Section 2 is devoted to the generalized Schur algorithm which is the basis of the superfast linear least squares Schur algorithm. In this Section, we start by describing the Cholesky factorization algorithm in subsection 2.1. The Cholesky factorization procedure is used to decompose a s.p.d matrix into a lower and upper triangular form. This subsection is followed by the subsection describing the displacement representation of the Toeplitz matrix and how we can deduce that its displacement rank is 4. Alongside, we explain what a generator matrix is as defined by the matrix G. Other concepts such as proper form reduction, signature matrix, downshift matrix and the zero-bordered Schur complement are introduced and explained also here. Subsection 2.3 is devoted towards the reduction of the matrix G to proper form.

In Section 3, we explain the superfast linear least squares Schur algorithm. In this Section, we define matrix generators, calculate the displacement rank of the regularized normal equations. We will also show how the displacement rank of the regularized matrix $T^TT+\alpha^2I$ can be reduced from 5 to 4. This is followed by the subsection which explains how we can factorize a regularized Toeplitz-like matrix. Other procedures discussed here include the computation of the generators of $M = T^TT + \alpha^2I$ and its subsequent application to finding the generators of M^{-1}. We end this Section with the fast matrix multiplication by a circulant matrix used to speed up the superfast linear least squares problem via the FFT process. In Section 4, we examine the complexity and storage analysis of the superfast linear least squares problem. This Section is immediately followed by Section 5 which dwells on the application of the superfast linear least squares algorithm to image deblurring. Section 6 presents the image restoration research results obtained from our experiments. This subsection is followed by a detailed analysis of the results. Additionally, in this Section we present a summary of the results found in the thesis research. Finally, Section 7 concludes our thesis

research work.

1.1 The Problem of Ill-conditioning

Given a general linear system $B = T_1XT_2$, the problem is to solve for X. Note that T_1, T_2 and X are the given data with some perturbations in them such as: rounding error, measurement error, variations in the signal capture, environmental distortions and more. If perturbation is present in the matrices used, then $\hat{T}_1 = T_1 + E_1$, $\hat{T}_2 = T_2 + E_2$ and $\hat{B} = B + F$. In the matrix B, F represents noise present in the blurred image. In our thesis work, the perturbations E_1 and E_2 present in matrices T_1 and T_2 are not part of our model of blurring, but we will include them in our perturbation analysis here since they represent backward errors in the solution of our linear system problem.

This implies that as a result of the perturbations, the system we need to solve for X becomes

$$(T_1 + E_1)\hat{X}(T_2 + E_2) = B + F$$

Since $T_1XT_2 = B$, one can express the equation above as

$$(T_1 + E_1)\hat{X}(T_2 + E_2) = T_1XT_2 + F$$

which on further expansion, we determine that

$$F = \left(T_1\hat{X} + E_1\hat{X}\right)(T_2 + E_2) - T_1XT_2.$$

Note that the same result can be expressed as

$$T_1\hat{X}T_2 + E_1\hat{X}T_2 + T_1\hat{X}E_2 + E_1\hat{X}E_2 - T_1XT_2 = F$$

and this is also equivalent to the equation

$$T_1\hat{X}T_2 - T_1XT_2 = F - \left(E_1\hat{X}T_2 + T_1\hat{X}E_2 + E_1\hat{X}E_2\right).$$

When we simplify the result above, we determine that

$$T_1\left(\hat{X} - X\right)T_2 = F - \left(E_1\hat{X}T_2 + T_1\hat{X}E_2 + E_1\hat{X}E_2\right).$$

Using this derived equation, one can approximate the error in the system being solved using the expression

$$\hat{X} - X = T_1^{-1}\left(F - \left(E_1\hat{X}T_2 + T_1\hat{X}E_2\right)\right)T_2^{-1} + \mathrm{O}\left(E_1\hat{X}E_2\right).$$

Clearly, the neglected second order term in the error approximation is

$$\mathrm{O}\left(E_1\hat{X}E_2\right)$$

Note from the calculations above that by the $2-$norm

$$\|\hat{X} - X\|_2 \leq \|T_1^{-1}\left(F - \left(E_1\hat{X}T_2 + T_1\hat{X}E_2\right)\right)T_2^{-1}\|_2 + \mathrm{O}\left(\|E_1\hat{X}E_2\|_2\right).$$

This further indicates that according to the triangle inequality

$$\begin{aligned}
\|\hat{X} - X\|_2 &\leq \|T_1^{-1}\left(F - \left(E_1\hat{X}T_2 + T_1\hat{X}E_2\right)\right)T_2^{-1}\|_2 \\
&\quad + \mathrm{O}\left(\|E_1\hat{X}E_2\|_2\right) \\
&\leq \|T_1^{-1}\|_2\|T_2^{-1}\|_2\|\left(F - \left(E_1\hat{X}T_2 + T_1\hat{X}E_2\right)\right)\|_2 \\
&\quad + \mathrm{O}\left(\|E_1\hat{X}E_2\|_2\right) \\
&\leq \|T_1^{-1}\|_2\|T_2^{-1}\|_2\|F\|_2 + \\
&\quad \|T_1^{-1}\|_2\|T_2^{-1}\|_2\|E_1\|_2\|\hat{X}\|_2\|T_2\|_2 + \\
&\quad \|T_1^{-1}\|_2\|T_2^{-1}\|_2\|T_1\|_2\|\hat{X}\|_2\|E_2\|_2 \\
&\quad + \mathrm{O}\left(\|E_1\hat{X}E_2\|_2\right).
\end{aligned}$$

The relations obtained from the triangle inequality indicates that if we define

$$\kappa(T_1) = \|T_1^{-1}\|_2\|T_1\|_2$$

and

$$\kappa(T_2) = \|T_2^{-1}\|_2\|T_2\|_2$$

the relative error in X satisfies the following condition

$$\begin{aligned}
\frac{\|\hat{X} - X\|_2}{\|\hat{X}\|_2} &\leq \kappa\left(T_1\right)\kappa\left(T_2\right)\frac{\|F\|_2}{\|T_2\|_2\|T_1\|_2\|\hat{X}\|_2} + \\
&\quad \kappa\left(T_1\right)\kappa\left(T_2\right)\frac{\|E_1\|_2}{\|T_1\|_2} + \kappa\left(T_1\right)\kappa\left(T_2\right)\frac{\|E_2\|_2}{\|T_2\|_2} + \mathrm{O}\left(\|E_1\hat{X}E_2\|_2\right).
\end{aligned}$$

The quantities $\kappa(T_1)$ and $\kappa(T_2)$ are called the condition numbers for T_1 and T_2 respectively. By definition, a condition number is a natural measure of the relative distance to singularity of the linear system being solved such that

$$\min_{\hat{T}_1 \text{singular}} \frac{\|\hat{T}_1 - T_1\|_2}{\|T_1\|_2} = \frac{1}{\kappa_2(T_1)}$$

The reciprocal of $\kappa(T_1)$ gives the relative distance to singularity of the matrix under consideration. If the condition number of the matrix in the system being solved is large, we define such a system as being ill conditioned which implies that either $\|T_1\|_2$ or $\|T_1^{-1}\|_2$ is large. When this is the case, we say the problem being solved is ill-posed.

Mathematically speaking, the quality of a linear system or a linear least squares problem may be poor if the matrix involved is nearly singular. Therefore, the condition number allows us to measure whether a matrix is singular or non-singular.

1.2 Minimizing Errors using Tikhonov Regularization

The Tikhonov regularization or otherwise called damped least squares algorithm is an algorithm that was developed by Philips [7] and Tikhonov [8]. The algorithm is aimed at minimizing the ill-conditioning discussed earlier. It is well known that solutions to ill-conditioned problems are very sensitive to noise, and regularization methods can be used to stabilize them. To do so, one has to impose additional conditions to the solution being solved. The question then arises: which additional conditions does one need to impose on the regularized solution?

The Tikhonov regularization procedure is used in our research effort to damp out errors in the linear problem we are solving. The key concept here is that given

$$T_1 X T_2 = B$$

we set $Y = XT_2$ and solve the regularized system which requires that we compute

$$\min_{\hat{Y}} \| \begin{pmatrix} T_1 \\ \alpha I \end{pmatrix} Y - \begin{pmatrix} B \\ 0 \end{pmatrix} \|_F^2$$

where $\alpha \geq 0$ is the regularization parameter. This means that we can compute for

$$Y = \left(T_1^T T_1 + \alpha^2 I\right)^{-1} T_1^T B.$$

Alongside, solving for

$$XT_2 = Y \Leftrightarrow T_2^T X^T = Y^T$$

and this fact allows for the formulation of another linear least squares problem

$$\min_{X^T} \| \begin{pmatrix} T_2^T \\ \alpha I \end{pmatrix} X^T - \begin{pmatrix} Y^T \\ 0 \end{pmatrix} \|_F^2.$$

In this minimization problem, it is important to note that I is the identity matrix of order n, and its normal equations are given by

$$\left(T_2^T T_2 + \alpha^2 I\right) X^T = T_2^T Y^T$$

Note that the regularization parameter $\alpha \geq 0$ in the least squares problem controls the degree of minimizing the solution norm $\|X^T\|_F$ relative to minimizing the residual norm

$$\|T_2^T X^T - Y^T\|_F^2.$$

To dampen out errors in the system, one has to choose α appropriately for the given blurring, noise model and image. The choice of α affects the reproduction of high frequency information such as noise error and sharp edges. Our goal here is to choose α large enough to damp out noise but not to eliminate image characteristics such as sharp edges. Therefore when we use a regularization tool such as Tikhonov regularization, we seek to suppress the noise without losing image details of interest to us.

1.3 Application of Hyperbolic Rotations

In order to decompose the matrix being solved into upper and lower triangular form in our algorithms, we will need to introduce zeros to a matrix or a vector appropriately. One of the procedures needed to do so is with the application of a hyperbolic rotation which is a matrix of the form

$$H = \frac{1}{\sqrt{1-\rho^2}} \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$$

with $|\rho| \geq 0$ and real. Hyperbolic rotations satisfy the relation

$$\begin{aligned} H^T \Sigma H &= H \Sigma H^T \\ &= \Sigma \end{aligned}$$

for

$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

As an example, if we apply the hyperbolic rotations to a general matrix $G^T \in \Re^{2\times 2}$ where

$$G^T = \begin{pmatrix} g_{11} & g_{12}^T \\ g_{21} & g_{22}^T \end{pmatrix}$$

our goal is to use hyperbolic rotations to zero the element g_{21} in G^T. The elements g_{11}, g_{12}, and g_{22} are scalar quantities. Therefore, to introduce a zero at location for element g_{12}, we multiply G^T by the hyperbolic rotation matrix H to yield

$$\begin{aligned} HG &= \frac{1}{\sqrt{1-\rho^2}} \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \begin{pmatrix} g_{11} & g_{12}^T \\ g_{21} & g_{22}^T \end{pmatrix} \\ &= \frac{1}{\sqrt{1-\rho^2}} \begin{pmatrix} g_{11}+\rho g_{21} & g_{12}^T + \rho g_{22} \\ \rho g_{11} + g_{21} & \rho g_{12}^T + g_{22} \end{pmatrix}. \end{aligned}$$

If we set

$$\rho g_{11} + g_{21} = 0 \Rightarrow \rho g_{11} = -g_{21}$$

hence

$$\rho = \frac{-g_{21}}{g_{11}}$$

in the product HG^T. Using the definition of ρ, the product

$$HG^T = \begin{pmatrix} \hat{g}_{11} & \hat{g}_{12}^T \\ 0 & \hat{g}_{22}^T \end{pmatrix}.$$

The implication is that using the hyperbolic rotation, one can successfully introduce zeros to the elements in a matrix or vector as shown. More details of this procedure can be found in [4]. In Section 2, we will show how this concept is applied to the generalized Schur algorithm.

1.4 Application of Plane Rotations

Another way to introduce zeros into a matrix or a vector is by using the plane rotations. Details of this procedure can be found in [3]. Given $w \in \Re^{2\times 1}$, $c = \cos(\theta)$ and $s = \sin(\theta)$, one can use plane rotations to introduce zeros also in a vector or matrix selectively. Plane rotations are also referred to as Givens rotations. As an example, suppose we wish to introduce a zero in the second element of the vector

$$w = \begin{pmatrix} x \\ y \end{pmatrix}.$$

To do so, one has to set

$$c = \frac{x}{\sqrt{x^2 + y^2}}$$

$$s = \frac{y}{\sqrt{x^2 + y^2}}$$

which in turn allows for the definition of the plane rotation matrix

$$Q = \begin{pmatrix} c & -s \\ s & c \end{pmatrix}.$$

With Q defined, one can easily deduce that this matrix is orthogonal since

$$QQ^T = Q^TQ = I.$$

To zero the second element of the matrix w, we simply have to compute the product

$$Q^Tw = \begin{pmatrix} \sqrt{x^2 + y^2} \\ 0 \end{pmatrix}.$$

It follows from the result Q^Tw that plane rotations can also be used to introduce zeros into a vector or a matrix. This concept is applied along with the hyperbolic rotations procedure in the generalized Schur algorithm discussed in the next Section.

2 The Generalized Schur Algorithm

In this Section we derive the generalized Schur algorithm. This algorithm is used to factorize a symmetric positive definite Toeplitz or Toeplitz-like matrix. The fact that the matrix $M = T^TT + \alpha^2 I$ is s.p.d for $T = T_1$ or $T = T_2$, this implies that we can apply the Cholesky factorization procedure so that

$$T^TT + \alpha^2 I \;\; = \;\; C^TC.$$

The decomposition dictates that the Toeplitz-like matrix can be expressed as a product of a lower and upper triangular matrices where the matrix C is the upper triangular part. The factorization process can be accomplished using the Schur algorithm which is known

to compute the Cholesky factors with a runtime of $\mathrm{O}(n^2)$. This method is implemented with hyperbolic rotations in a factored form also known to provide numerical stability to the solutions. In this Section we will focus our subject matter to the discussion of the Cholesky factorization procedure and the displacement structure of the symmetric positive definite Toeplitz matrix $T^T T$. We will also show how its displacement equation can be used to define the generator matrix G. Later in the Section, we will show how the matrix G can be reduced to proper form using both hyperbolic and plane rotations. It must be noted that the quest of reducing the generator matrix to proper form is to successively reveal a row of the Cholesky factor and to put the generators in a form in which the generators of the Schur complement can be obtained with just a shift. The computation of the Cholesky factor then proceeds recursively. When the procedure is complete, we will have obtained the Cholesky matrix C which is an upper triangular matrix.

2.1 The Cholesky Factorization Procedure

Given a symmetric positive definite matrix, where s_0 is scalar, we define the Toeplitz matrix, $T \in \Re^{n \times n}$ as

$$T = \begin{pmatrix} s_0 & t_1^T \\ t_1 & T_2 \end{pmatrix}.$$

The procedure we will describe here is not the typical Cholesky factorization procedure but can lead to the computation of the Cholesky factor, C, such that

$$\begin{aligned} T - \begin{pmatrix} \sqrt{s_0} \\ \frac{t_1}{\sqrt{s_0}} \end{pmatrix} \begin{pmatrix} \sqrt{s_0} & \frac{t_1^T}{\sqrt{s_0}} \end{pmatrix} &= \begin{pmatrix} s_0 & t_1^T \\ t_1 & T_2 \end{pmatrix} - \begin{pmatrix} s_0 & t_1^T \\ t_1 & \frac{t_1 t_1^T}{s_0} \end{pmatrix} \\ \begin{pmatrix} 0 & 0 \\ 0 & T_s \end{pmatrix} &= \begin{pmatrix} 0 & 0 \\ 0 & T_2 - \frac{t_1 t_1^T}{s_0} \end{pmatrix} \end{aligned}$$

It is important to note that

$$\begin{pmatrix} \sqrt{s_0} & \frac{t_1^T}{\sqrt{s_0}} \end{pmatrix}$$

is the first Cholesky row. If we scale the first column of the Schur complement, $T_2 - \frac{t_1 t_1^T}{s_0}$, then the result is the second row of the Cholesky factor. To reveal the rest of the Cholesky rows, the process is done recursively to yield the Cholesky matrix C. In general, if a matrix is symmetric positive definite as $T \in \Re^{n \times n}$ is, then we can obtain a Cholesky factorization from it to yield

$$T^T T = C^T C$$

where C is an upper triangular matrix. This fact holds true for both Toeplitz and Toeplitz-like matrices.

To decompose a general matrix into its upper triangular form by the Cholesky factorization procedure, one has to simply compute c_{ij} where $1 \leq i, j \leq n$. This recursive procedure is described above in this subsection. A more explicit way of computing the Cholesky triangle elements, c_{ij}, is shown by the following $\mathrm{O}(n^3)$ Cholesky factorization algorithm for a general matrix. More details of this procedure can be found in [3]. The elements are computed recursively as follows:

$$\begin{aligned} c_{ii} &= \sqrt{\left(t_{ii} - \sum_{k=1}^{i-1} (c_{ik})^2 \right)} \\ c_{ji} &= \left(t_{ji} - \sum_{k=1}^{i-1} \frac{(c_{jk} c_{ik})}{c_{ii}} \right). \end{aligned}$$

In the next Section, we will consider how the complexity of this algorithm can be reduced to $O(n^2)$ or $O(n \log^2 n)$ for a Toeplitz-like matrix.

2.2 The Displacement Representation of a Structured Matrix

For greater generality, we now consider a rectangular Toeplitz matrix, $T \in \Re^{m \times n}$

$$T = \begin{pmatrix} s_0 & s_1 & s_2 & \cdots & \cdots & s_{n-1} \\ s_{-1} & s_0 & s_1 & s_2 & \cdots & \vdots \\ s_{-2} & s_{-1} & s_0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & s_{-(m-n-1)} \\ \vdots & \ddots & \ddots & \ddots & \ddots & s_{-(m-n)} \\ s_{-(m-1)} & s_{-(n-2)} & \cdots & \cdots & s_{-1} & s_{-(m-n+1)} \end{pmatrix} \tag{4}$$

Using this matrix definition, we can find a reduced representation of the s.p.d Toeplitz-like matrix $T^T T$ and thus efficiently solve the least squares problem. To do so, the displacement equation defined as

$$T^T T - Z T^T T Z^T = G \Sigma G^T$$

is evaluated. In the displacement equation, the matrix G is called the generator matrix, the matrix Z is referred to as the downshift matrix while the matrix Σ is the signature matrix. The downshift matrix Z is defined as

$$Z = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & 0 & \ddots & 0 \\ \vdots & 1 & 0 & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 \end{pmatrix}$$

where its definition is in such a way that ones are on the subdiagonal and zeros elsewhere. On the other hand, the signature matrix is defined as

$$\Sigma = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

such that the negative and positive ones on the main diagonalare derived from the decomposition of the matrix displacement equation.

To show this fact, we partition T in two different ways so that

$$T = \begin{pmatrix} s_0 & t_1^T \\ t_{-1} & T_2 \end{pmatrix} = \begin{pmatrix} T_2 & \hat{t}_1 \\ \hat{t}_{-1}^T & s_{m-n} \end{pmatrix}.$$

In the partitioning, $T_2 \in \Re^{m-1\times n-1}$, s_0 is a scalar quantity while $t_1^T \in \Re^{1\times n-1}$and $t_{-1} \in \Re^{m-1\times 1}$ are both vectors. Equivalently, s_{m-n} is a scalar quantity but $\hat{t}_{-1}^T \in \Re^{m-1\times 1}$ and $\hat{t}_1 \in \Re^{1\times n-1}$ are vectors.

With T defined, we can then compute

$$\begin{aligned} T^T T &= \begin{pmatrix} s_0 & t_{-1}^T \\ t_1 & T_2^T \end{pmatrix} \begin{pmatrix} s_0 & t_1^T \\ t_{-1} & T_2 \end{pmatrix} \\ &= \begin{pmatrix} s_0^2 + t_{-1}^T t_{-1} & s_0 t_1^T + t_{-1}^T T_2 \\ t_1 s_0 + T_2^T t_{-1} & t_1 t_1^T + T_2^T T_2 \end{pmatrix}. \end{aligned}$$

Alternatively if we let

$$T = \begin{pmatrix} T_2 & \hat{t}_1 \\ \hat{t}_{-1}^T & s_{m-n} \end{pmatrix}$$

then

$$
\begin{aligned}
T^T T &= \begin{pmatrix} T_2^T & \hat{t}_{-1} \\ \hat{t}_1^T & s_{m-n} \end{pmatrix} \begin{pmatrix} T_2 & \hat{t}_1 \\ \hat{t}_{-1}^T & s_{m-n} \end{pmatrix} \\
&= \begin{pmatrix} T_2^T T_2 + \hat{t}_{-1}\hat{t}_{-1}^T & T_2^T \hat{t}_1 + \hat{t}_{-1} s_{m-n} \\ \hat{t}_1^T T_2 + s_{m-n}\hat{t}_{-1}^T & \hat{t}_1^T \hat{t}_1 + s_{m-n}^2 \end{pmatrix}.
\end{aligned}
$$

Now, using the displacement equation definition, it is easy to see that

$$
\begin{aligned}
T^T T - Z T^T T Z^T &= \begin{pmatrix} s_0^2 + t_{-1}^T t_{-1} & s_0 t_1^T + t_{-1}^T T_2 \\ t_1 s_0 + T_2^T t_{-1} & t_1 t_1^T + T_2^T T_2 \end{pmatrix} - \\
&\quad \begin{pmatrix} 0 & 0 \\ 0 & T_2^T T_2 + \hat{t}_{-1}\hat{t}_{-1}^T \end{pmatrix}.
\end{aligned}
$$

Algebraically, this equation is equivalent to

$$
\begin{aligned}
T^TT - ZT^TTZ^T &= \begin{pmatrix} s_0^2 + t_{-1}^T t_{-1} & s_0 t_1^T + t_{-1}^T T_2 \\ t_1 s_0 + T_2^T t_{-1} & t_1 t_1^T - \hat{t}_{-1}\hat{t}_{-1}^T \end{pmatrix} \\
&= \begin{pmatrix} s_0 \\ t_1 \end{pmatrix} \begin{pmatrix} s_0 & t_1^T \end{pmatrix} + \begin{pmatrix} t_{-1}^T t_{-1} & t_{-1}^T T_2 \\ T_2^T t_{-1} & -\hat{t}_{-1}\hat{t}_{-1}^T \end{pmatrix} \\
&= \begin{pmatrix} s_0 \\ t_1 \end{pmatrix} \begin{pmatrix} s_0 & t_1^T \end{pmatrix} - \begin{pmatrix} 0 \\ \hat{t}_{-1} \end{pmatrix} \begin{pmatrix} 0 & \hat{t}_{-1}^T \end{pmatrix} + \\
&\quad \begin{pmatrix} t_{-1}^T t_{-1} & t_{-1}^T T_2 \\ T_2^T t_{-1} & 0 \end{pmatrix} \\
&= \begin{pmatrix} s_0 \\ t_1 \end{pmatrix} \begin{pmatrix} s_0 & t_1^T \end{pmatrix} - \begin{pmatrix} 0 \\ \hat{t}_{-1} \end{pmatrix} \begin{pmatrix} 0 & \hat{t}_{-1}^T \end{pmatrix} + \\
&\quad \begin{pmatrix} \| t_{-1} \|_2 \\ \frac{T_2^T t_{-1}}{\|t_{-1}\|_2} \end{pmatrix} \begin{pmatrix} \| t_{-1} \|_2 & \left(\frac{T_2^T t_{-1}}{\|t_{-1}\|_2}\right)^T \end{pmatrix} - \\
&\quad \begin{pmatrix} 0 \\ \frac{T_2^T t_{-1}}{\|t_{-1}\|_2} \end{pmatrix} \begin{pmatrix} 0 & \left(\frac{T_2^T t_{-1}}{\|t_{-1}\|_2}\right)^T \end{pmatrix}.
\end{aligned}
$$

From the decomposition of the displacement equation we can define the generator matrix as

$$
\mathrm{G} = \begin{pmatrix} s_0 & \|t_{-1}\|_2 & 0 & 0 \\ t_1 & \frac{T_2^T t_{-1}}{\|t_{-1}\|_2} & \hat{t}_{-1} & \frac{T_2^T t_{-1}}{\|t_{-1}\|_2} \end{pmatrix}.
$$

Note from the decomposition of the displacement equation $T^TT - ZT^TTZ^T$ that we have conclusively deduced that the displacement rank is 4. The displacement rank is defined as the rank of the displacement equation $T^TT - ZT^TTZ^T = G\Sigma G^T$. When the displacement rank is significantly low as this one is, it alludes to the fact that we can decompose T^TT easily into C^TC using the Cholesky factorization procedure.

In general, we will work with any matrix $M = T^T T + \alpha^2 I$ such that

$$\begin{aligned} M - ZMZ^T &= \hat{G}\Sigma\hat{G}^T \\ &= \begin{pmatrix} a_0 & b_0 & c_0 & d_0 \\ a_1 & b_1 & c_1 & d_1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} a_0 & a_1^T \\ b_0 & b_1^T \\ c_0 & c_1^T \\ d_0 & d_1^T \end{pmatrix} \end{aligned}$$

where a_0, b_0, c_0, and d_0 are scalars while a_1, b_1, c_1, and d_1 are vectors. Note that the $\Sigma-$orthogonal matrix described here is a matrix H such that $H\Sigma H^T = \Sigma$ where $\Sigma = I \oplus -I$. The matrix

$$H = \frac{1}{\sqrt{1-|\rho|^2}} \begin{pmatrix} \gamma_1 & 0 & 0 & 0 \\ 0 & \gamma_2 & 0 & 0 \\ 0 & 0 & \gamma_3 & 0 \\ 0 & 0 & 0 & \gamma_4 \end{pmatrix} \begin{pmatrix} 1 & 0 & \rho & 0 \\ 0 & 1 & 0 & 0 \\ \rho & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

is defined in such a way that $|\gamma_1| = |\gamma_2| = |\gamma_3| = |\gamma_4| = 1$ and $0 \leq |\rho| < 1$. In our algorithms, we will set

$$\Sigma = I_2 \oplus -I_2.$$

The signature matrix Σ used here enables us to construct any $\Sigma-$orthogonal matrices from products of hyperbolic rotations and block diagonal orthogonal matrices. A block diagonal orthogonal matrix is of the form

$$\begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & c_2 & -s_2 \\ 0 & 0 & s_2 & c_2 \end{pmatrix}$$

where c_1 and c_2 are real and non-negative scalar quantities while s_1 and s_2 are assumed to

be real. These values satisfy the condition that

$$c_1^2 + |s_2|^2 \;\; = \;\; c_2^2 + |s_2|^2 = 1.$$

It is also instructive to note that any hyperbolic rotation used is of the form

$$\begin{pmatrix} \frac{1}{\sqrt{1-|\rho_1|^2}} & 0 & \frac{\rho_1}{\sqrt{1-|\rho_1|^2}} & 0 \\ 0 & \frac{1}{\sqrt{1-|\rho_2|^2}} & 0 & \frac{\rho_2}{\sqrt{1-|\rho_2|^2}} \\ \frac{\bar{\rho_1}}{\sqrt{1-|\rho_1|^2}} & 0 & \frac{1}{\sqrt{1-|\rho_1|^2}} & 0 \\ 0 & \frac{\bar{\rho_2}}{\sqrt{1-|\rho_2|^2}} & 0 & \frac{1}{\sqrt{1-|\rho_2|^2}} \end{pmatrix}.$$

The $\Sigma-$orthogonal matrix can be formed as a product of plane rotations, hyperbolic rotations and diagonal matrices such that diagonal matrices are defined by

$$\text{diag}\,(\gamma_1, \gamma_2, \gamma_3, \gamma_4)$$

and $|\gamma_1| = |\gamma_2| = |\gamma_3| = |\gamma_4| = 1$.

2.3 Reducing G to Proper Form with $\Sigma-$Orthogonal Transformations

The idea behind the generalized Schur algorithm is that given

$$G^T = \begin{pmatrix} a_0 & a_1^T \\ b_0 & b_1^T \\ c_0 & c_1^T \\ d_0 & d_1^T \end{pmatrix}$$

also defined as the generator matrix, one can use both hyperbolic and plane rotations to introduce zeros in the elements of G as discussed in our introductory material. As a result,

we have

$$
\begin{aligned}
G^T &= \begin{pmatrix} \frac{1}{\sqrt{1-\rho^2}} & 0 & \frac{\rho}{\sqrt{1-\rho^2}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\rho}{\sqrt{1-\rho^2}} & 0 & \frac{1}{\sqrt{1-\rho^2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & c_2 & -s_2 \\ 0 & 0 & s_2 & c_2 \end{pmatrix} \\
&\quad \begin{pmatrix} a_0 & a_1^T \\ b_0 & b_1^T \\ c_0 & c_1^T \\ d_0 & d_1^T \end{pmatrix} \\
&= \begin{pmatrix} \hat{a}_0 & \hat{a}_1^T \\ 0 & \hat{b}_1^T \\ 0 & \hat{c}_1^T \\ 0 & \hat{d}_1^T \end{pmatrix}
\end{aligned}
$$

When these complete set of operations are done to introduce zeros as shown above, we say that we have reduced the matrix into proper form. A matrix G^T that has been reduced to proper form has the leading element of each row except the first equal to zero.

Therefore, if given the generator matrix G and a matrix H that satisfies the relationship

$$H\Sigma H^T = \Sigma$$

then we can deduce that

$$
\begin{aligned}
(GH)\Sigma(GH)^T &= G(H\Sigma H^T)G^T \\
&= G\Sigma G^T.
\end{aligned}
$$

From this result, it suffices to conclude that GH are also generators of some Toeplitz-like

matrix M with H being an arbitrary Σ-orthogonal transformation. In general we will always choose H to put G in proper form.

Now let us define the zero bordered Schur complement used to recursively evaluate the other Cholesky factor rows to be

$$\begin{pmatrix} 0 & 0 \\ 0 & M_s \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & M_{22} - \mu_{21}\mu_0^{-1}\mu_{12}^T \end{pmatrix}$$

where

$$M = \begin{pmatrix} \mu_0 & \mu_{21}^T \\ \mu_{21} & M_{22} \end{pmatrix}.$$

In the definition of the matrix M, μ_0 is a scalar quantity, $\mu_{21}^T \in \Re^{1\times n-1}$, $\mu_{12} \in \Re^{n-1\times 1}$ and $M_{22} \in \Re^{n-1\times n-1}$. From the definition of M given, it is important to note that we are working with the general matrix M. Equivalently, if

$$m_1^T = \begin{pmatrix} \mu_0 & \mu_1 & \cdots & \mu_{n-1} \end{pmatrix}$$

is the first row of M, then we can denote

$$\begin{pmatrix} 0 & 0 \\ 0 & M_s \end{pmatrix} = M - \frac{1}{\mu_0} m_1 m_1^T.$$

Using the displacement equation definition, it means that we can express

$$\begin{aligned} \begin{pmatrix} 0 & 0 \\ 0 & M_s \end{pmatrix} - Z \begin{pmatrix} 0 & 0 \\ 0 & M_s \end{pmatrix} Z^T &= M - ZMZ^T - \frac{1}{\mu_0} m_1 m_1^T \\ &\quad + \frac{1}{\mu_0} Z \left(m_1 m_1^T \right) Z^T \end{aligned}$$

Clearly

$$\begin{pmatrix} 0 & 0 \\ 0 & M_s \end{pmatrix} - Z \begin{pmatrix} 0 & 0 \\ 0 & M_s \end{pmatrix} Z^T = \begin{pmatrix} 0 & 0 \\ 0 & G_s \Sigma G_s^T \end{pmatrix}$$

where

$$G_s \;=\; \begin{pmatrix} Z\hat{a} & \hat{b} & \hat{c} & \hat{d} \end{pmatrix}$$

are the proper form generators for M_s.

This illustrates that the zero-bordered Schur complement M_s inherits the displacement structure of M. Therefore the generators of M_s are easily determined by the proper form generators of $T^T T$. In a more general case, the above steps of the generalized Schur algorithm are repeated recursively on M_s with the generator matrix G_s to successively compute the subsequent rows of the Cholesky factor.

3 The Superfast Linear Least Squares Schur Algorithm

As stated earlier in the introductory material, the quest of the superfast linear least squares Schur algorithm is to speed up the computations of the generalized Schur algorithm. More importantly, we already understand that the basis of this algorithm is the generalized Schur algorithm. To speed up our computations, it was noted that the FFT process is used. This means that the FFT process plays a major role in the matrix-vector multiplication needed in solving our linear system of equations.

In this Section we start with subsection 3.1 which deals with the computation of the generators of $T^T T$. Once we compute the generators and show that it is of rank 5, we deduce in subsection 3.2 how we can reduce the rank of $\hat{G}$ from 5 to 4. In subsection 3.3, we show how we can represent the matrix $\hat{G}$ as polynomials. This is continued with a detailed procedure about how we can factorize the regularized system we seek to solve. Turning to subsection 3.4, we show how we can reduce the generator matrix to proper form.

Subsection 3.5 is devoted to the computation of the generators for the matrix M^{-1}. Finally in subsection 3.6, we show how the FFT process speeds up the computations involved in solving our regularized system.

3.1 Computing the Generators of $M = T^T T + \alpha^2 I$

The first step of the superfast linear least squares Schur algorithm applied to a regularized system is to obtain the four generators needed to represent the matrix to be factored. From the generalized Schur algorithm, it was proved that $T^T T - Z\left(T^T T\right) Z^T$ is of rank 4. It was further shown from the expansion of $T^T T - Z\left(T^T T\right) Z^T$ that one can obtain the generator matrix

$$G = \begin{pmatrix} s_0 & \|t_{-1}\|_2 & 0 & 0 \\ t_1 & \frac{T_2^T t_{-1}}{\|t_{-1}\|_2} & \hat{t}_{-1} & \frac{T_2^T t_{-1}}{\|t_{-1}\|_2} \end{pmatrix}.$$

Now if instead we use the Toeplitz-like matrix $M = T^T T + \alpha^2 I$ and apply the displacement operator to the regularized normal equations we have

$$\begin{aligned} (M) - Z(M) Z^T &= T^T T + \alpha^2 I - Z T^T T Z^T - Z\alpha^2 I Z^T \\ &= T^T T - Z T^T T Z^T + \alpha^2 I - Z\alpha^2 I Z^T. \end{aligned}$$

The result can further be expressed as

$$\begin{aligned} M - Z(M) Z^T &= G\Sigma G^T + \alpha^2 e_1 e_1^T \\ &= \hat{G}\hat{\Sigma}\hat{G}^T. \end{aligned}$$

In the displacement equation, the vector e_1 is the first standard basis vector and α is a scalar quantity which precisely is the regularization parameter. From the above result it appears that the displacement rank of the new generator matrix $\hat{G}$ is 5 instead of 4 deduced before.

This new result allows us to redefine the generator matrix as

$$\hat{G} = \begin{pmatrix} s_0 & \|t_{-1}\|_2 & \alpha & 0 & 0 \\ t_1 & \frac{T_2^T t_{-1}}{\|t_{-1}\|_2} & 0 & \hat{t}_{-1} & \frac{T_2^T t_{-1}}{\|t_{-1}\|_2} \end{pmatrix}$$

whose signature matrix is given by

$$\hat{\Sigma} = I_3 \oplus -I_2.$$

Clearly, we have shown that the column vectors of $\hat{G}^T$ are also the generators of $M = T^T T + \alpha^2 I$ so that by using the displacement equation formula, we can now assert that

$$M - Z(M) Z^T = \hat{G}\hat{\Sigma}\hat{G}^T.$$

With this idea in mind, one can apply this concept to solve a regularized least squares problem.

We define a regularized least squares minimization problem to be

$$\min_{\hat{X}} \| \begin{pmatrix} T \\ \alpha I \end{pmatrix} \hat{x} - \begin{pmatrix} y \\ 0 \end{pmatrix} \|_2^2$$

where I is the identity matrix of order n and the rest is the regularized normal equations for the least squares problem

$$\begin{aligned} \begin{pmatrix} T^T & \alpha I \end{pmatrix} \begin{pmatrix} T \\ \alpha I \end{pmatrix} \hat{x} &= \begin{pmatrix} T^T & \alpha I \end{pmatrix} \begin{pmatrix} y \\ 0 \end{pmatrix} \\ (T^T T + \alpha^2 I)\hat{x} &= T^T y. \\ M\hat{x} &= T^T y \end{aligned}$$

Since M is a symmetric and positive definite Toeplitz-like matrix, it means that it has a

Cholesky factor. Therefore

$$(M)\hat{x} = T^T y \Leftrightarrow \left(C^T C\right)\hat{x} = T^T y.$$

This yields a linear problem which can easily be solved by back substitution. The overall computational complexity for this linear probem can be deduced to be $\mathrm{O}(n^2)$.

3.2 Reducing the Displacement Rank from 5 to 4

Given the Toeplitz-like matrix

$$M = T^T T + \alpha^2 I$$

then by using the displacement equation definition, we can show that

$$\begin{aligned} M - ZMZ^T &= \begin{pmatrix} s_0 \\ t_1 \end{pmatrix}\begin{pmatrix} s_0 & t_1^T \end{pmatrix} - \begin{pmatrix} 0 \\ \hat{t}_{-1} \end{pmatrix}\begin{pmatrix} 0 & \hat{t}_{-1}^T \end{pmatrix} + \\ &\quad \begin{pmatrix} \| t_{-1} \|_2 \\ \frac{T_2^T t_{-1}}{\|t_{-1}\|_2} \end{pmatrix}\begin{pmatrix} \| t_{-1} \|_2 & \left(\frac{T_2^T t_{-1}^T}{\|t_{-1}\|_2}\right)^T \end{pmatrix} - \\ &\quad \begin{pmatrix} 0 \\ \frac{T_2^T t_{-1}}{\|t_{-1}\|_2} \end{pmatrix}\begin{pmatrix} 0 & \left(\frac{T_2^T t_{-1}^T}{\|t_{-1}\|_2}\right)^T \end{pmatrix} + \begin{pmatrix} \alpha \\ 0 \end{pmatrix}\begin{pmatrix} \alpha & 0 \end{pmatrix}. \end{aligned}$$

Note that the above equation is equivalent to

$$\begin{aligned} M - ZMZ^T &= \begin{pmatrix} s_0 \\ t_1 \end{pmatrix}\begin{pmatrix} s_0 & t_1^T \end{pmatrix} - \begin{pmatrix} 0 \\ \hat{t}_{-1} \end{pmatrix}\begin{pmatrix} 0 & \hat{t}_{-1}^T \end{pmatrix} + \\ &\quad \begin{pmatrix} \|t_{-1}\|_2^2 + \alpha^2 & t_{-1}^T T_2 \\ T_2^T t_{-1} & 0 \end{pmatrix} \end{aligned}$$

since $\| t_{-1} \|_2 = t_{-1}t_{-1}^T$. Continuing the expansion shows that

$$\begin{aligned} M - ZMZ^T &= \begin{pmatrix} s_0 \\ t_1 \end{pmatrix} \begin{pmatrix} s_0 & t_1^T \end{pmatrix} - \begin{pmatrix} 0 \\ \hat{t}_{-1} \end{pmatrix} \begin{pmatrix} 0 & \hat{t}_{-1}^T \end{pmatrix} + \\ & \begin{pmatrix} \sqrt{\|t_{-1}\|_2^2 + \alpha^2} \\ \frac{T_2^T t_{-1}}{\sqrt{\|t_{-1}\|_2^2+\alpha^2}} \end{pmatrix} \\ & \begin{pmatrix} \sqrt{\|t_{-1}\|_2^2 + \alpha^2} & \frac{t_{-1}^T T_2}{\sqrt{\|t_{-1}\|_2^2+\alpha^2}} \end{pmatrix} - \\ & \begin{pmatrix} 0 \\ \frac{T_2^T t_{-1}}{\sqrt{\|t_{-1}\|_2^2+\alpha^2}} \end{pmatrix} \begin{pmatrix} 0 & \frac{t_{-1}^T T_2}{\sqrt{\|t_{-1}\|_2^2+\alpha^2}} \end{pmatrix}. \end{aligned}$$

Clearly

$$M - ZMZ^T = \hat{G}\hat{\Sigma}\hat{G}^T$$

and more importantly, it is conclusive that the rank of $\hat{G}\hat{\Sigma}\hat{G}^T$ has been reduced from 5 to 4. This allows us to give a much more explicit definition for the column vectors of $\hat{G}$ as

$$\hat{G} = \begin{pmatrix} s_0 & \sqrt{\|t_{-1}\|_2^2 + \alpha^2} & 0 & 0 \\ t_1 & \frac{T_2^T t_{-1}}{\sqrt{\|t_{-1}\|_2^2+\alpha^2}} & \hat{t}_{-1} & \frac{T_2^T t_{-1}}{\sqrt{\|t_{-1}\|_2^2+\alpha^2}} \end{pmatrix}.$$

The new matrix $\hat{G}$ stores the generators of the regularized least squares linear problem. It is also instructive to note that the generators can be representated in the form of polynomials using the relation

$$\begin{aligned} \hat{G}(z) &= \begin{pmatrix} 1 & z & z^2 & \cdots & z^{n-1} \end{pmatrix} \begin{pmatrix} a_0 & b_0 & c_0 & d_0 \\ a_1 & b_1 & c_1 & d_1 \\ \vdots & \vdots & \vdots & \vdots \\ a_{n-1} & b_{n-1} & c_{n-1} & d_{n-1} \end{pmatrix} \\ &= \begin{pmatrix} a_0(z) & b_0(z) & c_0(z) & d_0(z) \end{pmatrix}. \end{aligned}$$

Here, the zero subscripts denote that

$$\begin{pmatrix} a_0(z) & b_0(z) & c_0(z) & d_0(z) \end{pmatrix}$$

are the initial generators. This representation combined with the fast multiplication of the polynomials via the FFT will further speed up the algorithm.

3.3 Factorizing the Regularized System

In subsection 3.2, we indicated that the matrix $\hat{G}$ can be represented as polynomial generators. This means that the generator matrix becomes

$$\hat{G} = \begin{pmatrix} a_0 & b_0 & c_0 & d_0 \\ a_1 & b_1 & c_1 & d_1 \\ \vdots & \vdots & \vdots & \vdots \\ a_{n-2} & b_{n-2} & c_{n-2} & d_{n-2} \\ a_{n-1} & b_{n-1} & c_{n-1} & d_{n-1} \end{pmatrix}$$

where the initial polynomial generator $\hat{G}$ is defined as

$$\hat{G}_0(z) = \begin{pmatrix} a_0(z) & b_0(z) & c_0(z) & d_0(z) \end{pmatrix}.$$

To perform the downshifts on the polynomial coefficients, one has to multiply the generator matrix with the downshift matrix

$$S(z) = \begin{pmatrix} z & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

on the right. These downshifting matrix is applied along with the plane and hyperbolic

rotations to recursively reveal the Cholesky rows needed to define the upper triangular Cholesky matrix C.

This implies after reducing the matrix into proper form using hyperbolic and plane rotations, the next step of the algorithm involves shifting the first column of G by multiplying with the shift matrix $S(z)$ so that

$$\begin{pmatrix} a_1(z) & b_1(z) & c_1(z) & d_1\ (z) \end{pmatrix} = \begin{pmatrix} a_0(z) & b_0(z) & c_0(z) & d_0\ (z) \end{pmatrix} \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & c_2 & -s_2 \\ 0 & 0 & s_2 & c_2 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{1-\rho^2}} & 0 & \frac{\rho}{\sqrt{1-\rho^2}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\rho}{\sqrt{1-\rho^2}} & 0 & \frac{1}{\sqrt{1-\rho^2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} z & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

When this is done, the new generator matrix is for the Schur complement. Continuing this

iteration, it implies that after k-steps, we have

$$\begin{pmatrix} a_k(z) & b_k(z) & c_k(z) & d_k\ (z) \end{pmatrix} = \begin{pmatrix} a_{k-1}(z) & b_{k-1}(z) & c_{k-1}(z) & d_{k-1}\ (z) \end{pmatrix} \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & c_2 & -s_2 \\ 0 & 0 & s_2 & c_2 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{1-\rho^2}} & 0 & \frac{\rho}{\sqrt{1-\rho^2}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\rho}{\sqrt{1-\rho^2}} & 0 & \frac{1}{\sqrt{1-\rho^2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} z & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The idea here is to reduce the polynomial generators into proper form after the plane and hyperbolic rotations have been applied so that

$$\begin{aligned} a(z) &= a_0 + a_1 z + a_2 z^2 + \cdots + a_{n-1} z^{n-1} \\ b(z) &= 0 + b_1 z + b_2 z^2 + \cdots + b_{n-1} z^{n-1} \\ c(z) &= 0 + c_1 z + c_2 z^2 + \cdots + c_{n-1} z^{n-1} \\ d(z) &= 0 + d_1 z + d_2 z^2 + \cdots + d_{n-1} z^{n-1}. \end{aligned}$$

In terms of matrices, this means that given the generator matrix

$$\hat{G}(z) = \begin{pmatrix} a(z) & b(z) & c(z) & d(z) \end{pmatrix}$$

where $a(z)$, $b(z)$, $c(z)$ and $d(z)$ are polynomials with n coefficients. Then the following simple steps are undertaken to reduce $\hat{G}(z)$ to proper form:

1. Apply hyperbolic and plane rotations to $\hat{G}(z)$ to yield

$$\hat{G}(z) = \begin{pmatrix} (\hat{a}_0 + \hat{a}_1 z + \cdots + \hat{a}_{n-1} z^{n-1}) & \left(\hat{b}_1 z + \cdots + \hat{b}_{n-1} z^{n-1}\right) \\ (\hat{c}_1 z + \cdots + \hat{c}_{n-1} z^{n-1}) & \left(\hat{d}_1 z + \cdots + \hat{d}_{n-1} z^{n-1}\right) \end{pmatrix}$$

2. The first Cholesky row is given by coefficients for

$$\begin{pmatrix} \hat{a_0} \\ \hat{a_1} \\ \hat{a}_2 \\ \vdots \\ \hat{a}_{n-1} \end{pmatrix}$$

3. Then next we apply the downshift matrix to

$$\begin{pmatrix} \hat{a_0} \\ \hat{a_1} \\ \hat{a}_2 \\ \vdots \\ \hat{a}_{n-1} \end{pmatrix}$$

so that

$$\tilde{G}(z) = \begin{pmatrix} (\hat{a}_0 + \hat{a}_1 z + \cdots + \hat{a}_{n-1} z^{n-1}) & \left(\hat{b}_1 z + \cdots + \hat{b}_{n-1} z^{n-1}\right) \\ (\hat{c}_1 z + \cdots + \hat{c}_{n-1} z^{n-1}) & \left(\hat{d}_1 z + \cdots + \hat{d}_{n-1} z^{n-1}\right) \end{pmatrix} \cdot S(z)$$

The new matrix $\tilde{G}$ is the generator matrix for the Schur complement after one complete

set of operations. We can define this new generator matrix as

$$\tilde{G}(z) = \begin{pmatrix} \tilde{a}(z) & \tilde{b}(z) & \tilde{c}(z) & \tilde{d}(z) \end{pmatrix}$$

4. Applying the same procedure to the Schur complement gives the second row of the Cholesky factor. The process continues recursively to reveal the rest of the Cholesky rows needed to define the Cholesky matrix, C.

Having reduced all the polynomial generators into proper form, the superfast linear least squares Schur algorithm can then be reformulated to use matrix-vector polynomial multiplication which has computational speedup via FFT. In this algorithm, given M, one can partition it so that we can take advantage of the divide and conquer technique. The partitioning is performed as follows

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{12}^T & M_{22} \end{pmatrix}$$

where $M_{11} \in \Re^{\frac{n}{2}\times\frac{n}{2}}$, $M_{12} \in \Re^{\frac{n}{2}\times\frac{n}{2}}$, $M_{12}^T \in \Re^{\frac{n}{2}\times\frac{n}{2}}$ and $M_{22} \in \Re^{\frac{n}{2}\times\frac{n}{2}}$. Given the partitioning, the transformations that processes k-steps of the superfast least squares Schur algorithm is of the form

$$\begin{pmatrix} a_k(z) & b_k(z) & c_k(z) & d_k(z) \end{pmatrix} = \begin{pmatrix} a_0(z) & b_0(z) & c_0(z) & d_0\ (z) \end{pmatrix} H_1 S(z) H_2 S(z) \cdots H_k S(z).$$

In this case, the initial generator polynomials are $a_0(z)$, $b_0(z)$, $c_0(z)$, $d_0(z)$ while the $a_k(z)$, $b_k(z)$, $c_k(z)$, $d_k(z)$ are the polynomial generators at the k^{th} step. The matrices $H_{1\cdots k}$ represents the products of hyperbolic and plane rotations while $S_{1\cdots k}$ represents the downshift matrices applied iteratively to arrive at the k^{th} step. The same result can be obtained by

the operation

$$\begin{pmatrix} a_k(z) & b_k(z) & c_k(z) & d_k(z) \end{pmatrix} = \begin{pmatrix} a_0(z) & b_0(z) & c_0(z) & d_0(z) \end{pmatrix} \begin{pmatrix} h_{11}^{(k)}(z) & h_{12}^{(k)}(z) & h_{13}^{(k)}(z) & h_{14}^{(k)}(z) \\ h_{21}^{(k)}(z) & h_{22}^{(k)}(z) & h_{23}^{(k)}(z) & h_{24}^{(k)}(z) \\ h_{31}^{(k)}(z) & h_{32}^{(k)}(z) & h_{33}^{(k)}(z) & h_{34}^{(k)}(z) \\ h_{41}^{(k)}(z) & h_{42}^{(k)}(z) & h_{43}^{(k)}(z) & h_{44}^{(k)}(z) \end{pmatrix}$$

More generally, let $H_j^{(k)}(z)$ be the matrix that performs k steps of the superfast linear least squares Schur algorithm on the step j generators defined as

$$\begin{pmatrix} a_j(z) & b_j(z) & c_j(z) & d_j(z) \end{pmatrix}.$$

Then it follows that

$$H_j^{(k)}(z) = H_j^{(\frac{k}{2})}(z) H_{j+\frac{k}{2}}^{(\frac{k}{2})}(z).$$

This is the single most important step in the algorithm. It is the doubling relation which permits the divide-and-conquer feature in the sflsschur algorithm. In particular, let $a_j^{(l)}(z)$, $b_j^{(l)}(z)$, $c_j^{(l)}(z)$ and $d_j^{(l)}(z)$ be generators after j steps truncated to be of length l.

This implies that the sflsschur algorithm is seen to admit the following recursive algorithm

function $[H_0^{(n)}(z)] = \text{sflsschur}(a_0^{(n)}(z), b_0^{(n)}(z), c_0^{(n)}(z), d_0^{(n)}(z))$

$$\begin{aligned} H_0^{(\frac{n}{2})}(z) &= \text{sflsschur}(a_0^{(\frac{n}{2})}(z), b_0^{(\frac{n}{2})}(z), c_0^{(\frac{n}{2})}(z), d_0^{(\frac{n}{2})}(z)) \\ \begin{pmatrix} a_{\frac{n}{2}}(z) & b_{\frac{n}{2}}(z) & c_{\frac{n}{2}}(z) & d_{\frac{n}{2}}(z) \end{pmatrix} &= \begin{pmatrix} a_0^{(n)}(z) & b_0^{(n)}(z) & c_0^{(n)}(z) & d_0^{(n)}(z) \end{pmatrix} H_0^{(\frac{n}{2})}(z) \\ H_{\frac{n}{2}}^{(\frac{n}{2})}(z) &= \text{sflsschur}(a_{\frac{n}{2}}^{(\frac{n}{2})}(z), b_{\frac{n}{2}}^{(\frac{n}{2})}(z), c_{\frac{n}{2}}^{(\frac{n}{2})}(z), d_{\frac{n}{2}}^{(\frac{n}{2})}(z)) \\ H_0^{(n)}(z) &= H_0^{(\frac{n}{2})}(z) H_{\frac{n}{2}}^{(\frac{n}{2})}(z) \end{aligned}$$

end

Evidently, the first step of the algorithm generates the Schur transformation matrix $H_0^{(\frac{n}{2})}(z)$ which corresponds to performing the Schur algorithm on M_{11}. This is defined by the function call

$$H_0^{(\frac{n}{2})}(z) = \text{sflsschur}(a_0^{(\frac{n}{2})}(z), b_0^{(\frac{n}{2})}(z), c_0^{(\frac{n}{2})}(z), d_0^{(\frac{n}{2})}(z)).$$

The procedure call evaluates the generators for the Schur complement $M_s = M_{22} - M_{12}^T M_{11}^{-1} M_{12}$. In the next step, we continue the factorization on M_s given that

$$\begin{pmatrix} a_{\frac{n}{2}}(z) & b_{\frac{n}{2}}(z) & c_{\frac{n}{2}}(z) & d_{\frac{n}{2}}(z) \end{pmatrix} = \begin{pmatrix} a_0^{(n)}(z) & b_0^{(n)}(z) & c_0^{(n)}(z) & d_0^{(n)}(z) \end{pmatrix} H_0^{(\frac{n}{2})}(z)$$

The factorization of M_s is such that

$$H_{\frac{n}{2}}^{(\frac{n}{2})}(z) = \text{sflsschur}(a_{\frac{n}{2}}^{(\frac{n}{2})}(z), b_{\frac{n}{2}}^{(\frac{n}{2})}(z), c_{\frac{n}{2}}^{(\frac{n}{2})}(z), d_{\frac{n}{2}}^{(\frac{n}{2})}(z)).$$

When we combine the first $\frac{n}{2}$ steps of the Schur algorithm with the last $\frac{n}{2}-$steps, this yields $H_0^{(n)}(z)$ or more equivalently

$$H_0^{(n)}(z) = H_0^{(\frac{n}{2})}(z) H_{\frac{n}{2}}^{(\frac{n}{2})}(z)$$

where

$$H = \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & c_2 & -s_2 \\ 0 & 0 & s_2 & c_2 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{1-\rho^2}} & 0 & \frac{\rho}{\sqrt{1-\rho^2}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\rho}{\sqrt{1-\rho^2}} & 0 & \frac{1}{\sqrt{1-\rho^2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} z & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Using the sflsschur algorithm, the problem is halved in each step such that when $n = 1$,

which corresponds to the last step of the Schur algorithm applied to

$$\begin{pmatrix} \sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 \end{pmatrix}$$

where σ_1, σ_2, σ_3, σ_4 are scalar quantities, we can reduce the matrix into

$$\begin{pmatrix} \tilde{\sigma} & 0 & 0 & 0 \end{pmatrix}$$

To perform the transformation, we use both hyperbolic and plane rotations. In particular we seek to find c_1, c_2, s_1, s_2 and ρ so that

$$\begin{pmatrix} \tilde{\sigma} & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} \sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 \end{pmatrix} \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & c_2 & -s_2 \\ 0 & 0 & s_2 & c_2 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{1-\rho^2}} & 0 & \frac{\rho}{\sqrt{1-\rho^2}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\rho}{\sqrt{1-\rho^2}} & 0 & \frac{1}{\sqrt{1-\rho^2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

thus yielding the generators in proper form.

3.4 The Transformation to Proper Form

When $n = 1$, we have the matrix

$$\begin{pmatrix} \sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 \end{pmatrix}$$

where the elements σ_1, σ_2, σ_3, σ_4 are scalar quantities and real. We need to establish here that

$$\begin{pmatrix} \tilde{\sigma} & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} \sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 \end{pmatrix} \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & c_2 & -s_2 \\ 0 & 0 & s_2 & c_2 \end{pmatrix}$$
$$\begin{pmatrix} \frac{1}{\sqrt{1-\rho^2}} & 0 & \frac{\rho}{\sqrt{1-\rho^2}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\rho}{\sqrt{1-\rho^2}} & 0 & \frac{1}{\sqrt{1-\rho^2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Proof:

Recall from [4] that using the plane rotations, also called givens rotations, we can reduce the matrix

$$\begin{pmatrix} a & b \end{pmatrix}$$

by the operation

$$\begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} c & -s \\ s & c \end{pmatrix} = \begin{pmatrix} r & 0 \end{pmatrix}$$

where $c = \frac{a}{\sqrt{a^2+b^2}}$ and $s = \frac{b}{\sqrt{a^2+b^2}}$. When we do so, we denoted that the result is in proper form. Applying the concept to the matrix product given in this problem, we start by setting

$$\begin{pmatrix} c_1 \\ s_1 \end{pmatrix} = \frac{\begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}}{\sqrt{\sigma_1^2 + \sigma_2^2}}$$

and

$$\begin{pmatrix} c_2 \\ s_2 \end{pmatrix} = \frac{\begin{pmatrix} \sigma_3 \\ \sigma_4 \end{pmatrix}}{\sqrt{\sigma_3^2 + \sigma_4^2}}.$$

Equivalently, if we multiply out the right hand side with the new definitions of c_1, s_1, c_2 and s_2 then

$$\begin{pmatrix} \hat{\sigma}_1 & 0 & \hat{\sigma}_3 & 0 \end{pmatrix} = \begin{pmatrix} \sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 \end{pmatrix} \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & c_2 & -s_2 \\ 0 & 0 & s_2 & c_2 \end{pmatrix}.$$

This means that if we multiply out the first part of the matrix, we can show that

$$\begin{aligned} \hat{\sigma}_1 &= \frac{\sigma_1^2 + \sigma_2^2}{\sqrt{\sigma_1^2 + \sigma_2^2}} \\ &= \sqrt{\sigma_1^2 + \sigma_2^2} \end{aligned}$$

and

$$\begin{aligned} \hat{\sigma}_3 &= \frac{\sigma_3^2 + \sigma_4^2}{\sqrt{\sigma_3^2 + \sigma_4^2}} \\ &= \sqrt{\sigma_3^2 + \sigma_4^2}. \end{aligned}$$

Alongside, if we let

$$\begin{aligned} \rho &= \frac{-\hat{\sigma}_3}{\hat{\sigma}_1} \\ &= \frac{\sqrt{\sigma_3^2 + \sigma_4^2}}{\sqrt{\sigma_1^2 + \sigma_2^2}} \end{aligned}$$

it is easy to see that completing the multiplication

$$\begin{aligned}
\begin{pmatrix} \hat{\sigma}_1 & 0 & \hat{\sigma}_3 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{1-\rho^2}} & 0 & \frac{\rho}{\sqrt{1-\rho^2}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\rho}{\sqrt{1-\rho^2}} & 0 & \frac{1}{\sqrt{1-\rho^2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} &= \begin{pmatrix} \left(\frac{\hat{\sigma}_1+\hat{\sigma}_3\rho}{\sqrt{1-\rho^2}}\right) & 0 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} \left(\frac{\hat{\sigma}_1+\hat{\sigma}_3\rho}{\sqrt{1-\rho^2}}\right) & 0 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} \left(\frac{\hat{\sigma}_1+\hat{\sigma}_3\left(\frac{-\hat{\sigma}_3}{\hat{\sigma}_1}\right)}{\sqrt{1-\left(\frac{-\hat{\sigma}_3}{\hat{\sigma}_1}\right)^2}}\right) & 0 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} \left(\frac{\frac{\hat{\sigma}_1^2-\hat{\sigma}_3^2}{\hat{\sigma}_1}}{\sqrt{\frac{\hat{\sigma}_1^2-\hat{\sigma}_3^2}{\hat{\sigma}_1^2}}}\right) & 0 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} \frac{\hat{\sigma}_1^2-\hat{\sigma}_3^2}{\sqrt{\sigma_1^2-\hat{\sigma}_3^2}} & 0 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} \sqrt{\hat{\sigma}_1^2-\hat{\sigma}_3^2} & 0 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} \tilde{\sigma} & 0 & 0 & 0 \end{pmatrix}
\end{aligned}$$

which is the proper form for the case when $n = 1$. Thus,

$$\tilde{\sigma} = \sqrt{(\sigma_1^2+\sigma_2^2)-\sqrt{\sigma_3^2+\sigma_4^2}}$$

3.5 Computation of the Generators of M^{-1}

Now recall that if given

$$T_1 X T_2 = B$$

to solve for X, we set $Y = XT_2$ and solve the regularized system using the Tikhonov regularization procedure which requires that we compute

$$\min_{\hat{Y}} \| \begin{pmatrix} T_1 \\ \alpha I \end{pmatrix} Y - \begin{pmatrix} B \\ 0 \end{pmatrix} \|_F^2$$

where $\alpha \geq 0$ is the regularization parameter. The normal regularized equations associated with the minimization problem is given by

$$Y = \left(T_1^T T_1 + \alpha^2 I\right)^{-1} T_1^T B.$$

Note that it is essential for us to compute for $M^{-1} = \left(T_1^T T_1 + \alpha^2 I\right)^{-1}$. Alongside, solving for

$$XT_2 = Y \Leftrightarrow T_2^T X^T = Y^T$$

allows for the formulation of another linear least squares problem

$$\min_{X^T} \| \begin{pmatrix} T_2^T \\ \alpha I \end{pmatrix} X^T - \begin{pmatrix} Y^T \\ 0 \end{pmatrix} \|_F^2.$$

Note that I is the identity matrix of order n. The other associated regularized normal equations are given by

$$\begin{aligned} \left(T_2^T T_2 + \alpha^2 I\right) X^T &= T_2^T Y^T \\ X^T &= \left(T_2^T T_2 + \alpha^2 I\right)^{-1} T_2^T Y^T \end{aligned}$$

which means we must evaluate for $M^{-1} = \left(T_2^T T_2 + \alpha^2 I\right)^{-1}$ using the Tikhonov regularization procedure. The regularization parameter $\alpha \geq 0$ in the least squares problem controls the

degree of minimizing the solution norm $\|X^T\|_F$ relative to minimizing the residual norm

$$\|T_2^T X^T - Y^T\|_F^2.$$

Therefore, if we define the Toeplitz-like matrix $M \in \Re^{n\times n}$ and $I \in \Re^{n\times n}$ so that

$$M = T^T T + \alpha^2 I$$

we can form the displacement equation for an augmented matrix as

$$\begin{pmatrix} M & I \\ I & 0 \end{pmatrix} - \begin{pmatrix} Z & 0 \\ 0 & Z \end{pmatrix} \begin{pmatrix} M & I \\ I & 0 \end{pmatrix} \begin{pmatrix} Z^T & 0 \\ 0 & Z^T \end{pmatrix} = \begin{pmatrix} M - ZMZ^T & e_1 e_1^T \\ e_1 e_1^T & 0 \end{pmatrix}$$

Note that the matrix

$$\begin{pmatrix} M & I \\ I & 0 \end{pmatrix}$$

yields the Schur complement

$$0 - IM^{-1}I = -M^{-1}.$$

This clearly illustrates that the Schur complement can be used to evaluate the matrix inverse M^{-1}. We can now define the displacement equation used to find the M^{-1} as

$$M^{-1} - ZM^{-1}Z^T = \hat{G}\hat{\Sigma}\hat{G}^T$$

where

$$\begin{aligned} \hat{G} &= \begin{pmatrix} a & b & c & d \end{pmatrix} \\ &= \begin{pmatrix} s_0 & \gamma & 0 & 0 \\ t_1 & \frac{T_2^T t_{-1}}{\gamma} & \hat{t}_{-1} & \frac{T_2^T t_{-1}}{\gamma} \end{pmatrix} \end{aligned}$$

and

$$\gamma = \sqrt{\|t_{-1}\|_2^2 + \alpha^2}.$$

It should be clear to the reader that the column vectors of $\hat{G}$ are also the generators of M. More equivalently

$$\begin{aligned}
\gamma e_1 &= \hat{G} \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \end{pmatrix} \\
&= \begin{pmatrix} \gamma \\ \frac{T_2^T t_{-1}}{\gamma} \end{pmatrix} - \begin{pmatrix} 0 \\ \frac{T_2^T t_{-1}}{\gamma} \end{pmatrix} \\
&= \begin{pmatrix} \gamma \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \end{pmatrix} \\
&= b - d.
\end{aligned}$$

Therefore, from the displacement equation of the augmented matrix, we evaluate that

$$\begin{aligned}
\begin{pmatrix} M & I \\ I & 0 \end{pmatrix} - \begin{pmatrix} Z & 0 \\ 0 & Z \end{pmatrix} \begin{pmatrix} M & I \\ I & 0 \end{pmatrix} \begin{pmatrix} Z^T & 0 \\ 0 & Z^T \end{pmatrix} &= \begin{pmatrix} M - ZMZ^T & e_1 e_1^T \\ e_1 e_1^T & 0 \end{pmatrix} \\
&= \begin{pmatrix} \hat{G}\hat{\Sigma}\hat{G}^T & e_1 e_1^T \\ e_1 e_1^T & 0 \end{pmatrix} \\
\begin{pmatrix} a & b & c & d \\ 0 & \frac{e_1}{\gamma} & 0 & \frac{e_1}{\gamma} \end{pmatrix} \Sigma \begin{pmatrix} a^T & 0 \\ b^T & \frac{e_1^T}{\gamma} \\ c^T & 0 \\ d^T & \frac{e_1^T}{\gamma} \end{pmatrix} &= \begin{pmatrix} \hat{G}\hat{\Sigma}\hat{G}^T & \frac{(b-d)e_1}{\gamma} \\ \frac{e_1(b-d)^T}{\gamma} & 0 \end{pmatrix} \\
&= \begin{pmatrix} \hat{G}\hat{\Sigma}\hat{G}^T & e_1 e_1^T \\ e_1 e_1^T & 0 \end{pmatrix}.
\end{aligned}$$

It follows then that the generators of the Schur complement of

$$\begin{pmatrix} M & I \\ I & 0 \end{pmatrix} = 0 - IM^{-1}I = -M^{-1}.$$

are

$$\begin{pmatrix} a & b & c & d \\ 0 & \frac{e_1}{\gamma} & 0 & \frac{e_1}{\gamma} \end{pmatrix} \in \Re^{2n \times 4}.$$

Recall from Calculus that

$$\begin{aligned}
(1-\alpha)\sum_{k=0}^{n} \alpha^j & \\
&= (1-\alpha)\left(1 + \alpha + \alpha^2 + \alpha^3 + \cdots + \alpha^{n-1} + \alpha^n\right) \\
&= \left(1 + \alpha + \alpha^2 + \cdots + \alpha^{n-1} + \alpha^n\right) - \left(\alpha + \alpha^2 + \cdots + \alpha^n + \alpha^{n+1}\right) \\
&= 1 - \alpha^{n+1}
\end{aligned}$$

Hence,

$$\begin{aligned}(1-\alpha)\sum_{k=0}^{n}\alpha^j &= 1-\alpha^{n+1}\\ \sum_{k=0}^{n}\alpha^j &= \frac{1-\alpha^{n+1}}{1-\alpha}\end{aligned}$$

Similarly, to compute the generators of M^{-1}, one can use the following relations

$$\begin{aligned}\sum_{k=0}^{n-1} Z^j\left(M-ZMZ^T\right)Z^{jT} &= \left(M-ZMZ^T\right)+Z\left(M-ZMZ^T\right)Z^T+\cdots\\ &+ Z^{n-1}\left(M-ZMZ^T\right)Z^{(n-1)T}\\ &= M\end{aligned}$$

Thus

$$\begin{aligned}M &= \sum_{k=0}^{n-1} Z^j\left(\hat{G}\hat{\Sigma}\hat{G}^T\right)Z^{jT}\\ &= \sum_{k=0}^{n-1} Z^j\left(\begin{pmatrix} a & b & c & d\end{pmatrix}\hat{\Sigma}\begin{pmatrix} a^T\\ b^T\\ c^T\\ d^T\end{pmatrix}\right)Z^{jT}\\ &= \sum_{k=0}^{n-1} Z^j\left(aa^T+bb^T-cc^T-dd^T\right)Z^{jT}\\ &= AA^T+BB^T-CC^T-DD^T\end{aligned}$$

where

$$A=\begin{pmatrix} a & Za & Z^2a & \cdots & Z^{n-1}a\end{pmatrix}.$$

The matrices B, C, and D are defined similarly. Note that A, B, C, and D are lower triangular Toeplitz matrices which are also the generators of M. The matrix M is shown to

be represented by the sum and differences of products of lower and upper triangular Toeplitz matrices. This fact is a generalization of the Gohberg-Semencul formula for the inverse of a Toeplitz matrix. Since each of the A, B, C, and D can be embedded in a $2n \times 2n$ circulant matrix, it is clear that we can perform fast multiplication by M via the FFT. It is important to note that after we run n steps of the superfast least squates Schur algorithm, we should obtain the generators of the matrix M^{-1}. Recall that if the generators are made polynomials, then e corresponds to the constant polynomial 1 such that

$$K(z) = \begin{pmatrix} a(z) & b(z) & c(z) & d(z) \\ 0 & \frac{1}{\gamma}(1) & 0 & \frac{1}{\gamma}(1) \end{pmatrix}$$

are the polynomial generators for the augmented matrix. This means that if $H(z)$ does n-steps of the superfast least squares Schur algorithm with the operation

$$\begin{pmatrix} a(z) & b(z) & c(z) & d(z) \\ 0 & \frac{1}{\gamma}(1) & 0 & \frac{1}{\gamma}(1) \end{pmatrix} H(z)$$

then it implies that after n-steps, we have the polynomial generators of the augmented matrix

$$\begin{pmatrix} 0 & 0 \\ 0 & -M^{-1} + ZM^{-1}Z^T \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & -M^{-1} \end{pmatrix} - \begin{pmatrix} Z & 0 \\ 0 & Z \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & -M^{-1} \end{pmatrix} \cdot \begin{pmatrix} Z^T & 0 \\ 0 & Z^T \end{pmatrix}$$

To calculate these polynomial generators, the equation

$$\begin{pmatrix} a_i(z) & b_i(z) & c_i(z) & d_i(z) \end{pmatrix} = \frac{1}{\gamma}\begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}\cdots \begin{pmatrix} h_{11}(z) & h_{12}(z) & h_{13}(z) & h_{14}(z) \\ h_{21}(z) & h_{22}(z) & h_{23}(z) & h_{24}(z) \\ h_{31}(z) & h_{32}(z) & h_{33}(z) & h_{34}(z) \\ h_{41}(z) & h_{42}(z) & h_{43}(z) & h_{44}(z) \end{pmatrix}$$

is used. This product evaluates the generators of M^{-1} and is equivalent to the matrix

$$\frac{1}{\gamma}\begin{pmatrix} h_{21}(z)+h_{41}(z) & h_{22}(z)+h_{42}(z) & h_{23}(z)+h_{43}(z) & h_{24}(z)+h_{44}(z) \end{pmatrix}.$$

3.6 Fast Multiplication Using FFT

Fast matrix-vector multiplication with a Toeplitz matrix can be performed at a time complexity of $O(n \log n)$ flops as opposed to $O(n^2)$ flops common for a general matrix vector multiplication. More details of this fact can be found in [2]. The superfast least squares Schur algorithm uses the FFT algorithm to implement fast matrix-vector multiplication and thus offers a clear computational speedup. This process works by embedding a $n \times n$ Toeplitz matrix into a large $2n \times 2n$ circulant matrix C so that the vectors are padded with zeros as highlighted in the process of computing the generators of M^{-1}. Given the superfast least squares Schur polynomials, k-steps of the superfast least squares Schur algorithm via polynomial multiplication can be performed. See details in [4]. The Schur polynomials can be calculated using a divide and conquer approach which is based on the doubling relations as discussed in the generalized Schur algorithm.

It should be noted that the FFT process cannot be used directly to evaluate the matrix-vector product. Instead, the Toeplitz matrix is embedded in a circulant matrix for which the FFT can be used to evaluate the circulant matrix-vector product. A $2n \times 2n$ matrix is

called circulant if it has the form

$$C = \begin{pmatrix} c_0 & c_{2n-1} & c_{2n-2} & \cdots & c_1 \\ c_1 & c_0 & c_{2n-1} & \cdots & c_2 \\ c_2 & c_1 & c_0 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & c_{2n-1} \\ c_{2n-1} & c_{2n-1} & \cdots & c_1 & c_0 \end{pmatrix}.$$

Note that the matrix C is a special kind of Toeplitz matrix where each column is obtained by doing a wrap-around downshift of the previous column. One can clearly see that the circulant matrix C has n elements satisfying the relation $c_{ij} = c_{(i-j)\mathrm{mod}n}$.

As an example on how we can construct a circulant matrix from a Toeplitz matrix, let T be a 3×3 Toeplitz matrix defined as

$$T = \begin{pmatrix} t_1 & 0 & 0 \\ t_2 & t_1 & 0 \\ t_3 & t_2 & t_1 \end{pmatrix}.$$

This means that the first column vector of the circulant matrix denoted by c is

$$c = \begin{pmatrix} t_1 & t_2 & t_3 & 0 & 0 & 0 \end{pmatrix}^T$$

with the extra zeros padded in it to make its dimensions $2n \times 2n$. Having embedded the

Toeplitz matrix T into the circulant matrix the result becomes

$$C = \begin{pmatrix} t_1 & 0 & 0 & 0 & t_3 & t_2 \\ t_2 & t_1 & 0 & 0 & 0 & t_3 \\ t_3 & t_2 & t_1 & 0 & 0 & 0 \\ 0 & t_3 & t_2 & t_1 & 0 & 0 \\ 0 & 0 & t_3 & t_2 & t_2 & 0 \\ 0 & 0 & 0 & t_3 & t_2 & t_1 \end{pmatrix}$$

This technique is completely general and can be applied to any $n \times n$ Toeplitz matrix. One can easily recognize the Toeplitz matrix T as the leading 3×3 principal submatrix of C. Since

$$C = \begin{pmatrix} T & S \\ S & T \end{pmatrix}$$

then in order to evaluate the linear equation $y = Tx$, one has to form a longer vector of the same dimension as the vector c so that

$$\bar{x} = \begin{pmatrix} x \\ 0 \end{pmatrix}$$

and thus we can obtain the matrix y as the top n elements in the result $\bar{y} = C\bar{x}$.

From above, note that the resulting circulant matrix can easily be diagonalized by the fourier transformation matrix,

$$F_{2n} = \frac{1}{\sqrt{(2n)}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \omega^2 & \cdots & \omega^{(2n)-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2((2n)-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{2n-1} & \omega^{2((2n)-1)} & \cdots & \omega^{2((2n)-1)\times 2((2n)-1)} \end{pmatrix}$$

which is unitary since $F_{2n}^{-1} = F_{2n}^{H}$ and $\omega = e^{\frac{-2\pi i}{2n}}$. It is a well known fact that a circulant matrix can be easily decomposed into

$$C = F_{2n}\Omega F_{2n}^{-1}$$

where the matrix F_{2n} is the Fourier matrix and Ω is the diagonal matrix whose elements are the Fourier transform of the first column of C [6].

3.6.1 The Divide and Conquer FFT Algorithm

Given t_0, t_1, t_2, $\cdots$, t_{n-1} elements of the Toeplitz matrices, one can compute

$$y_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} \omega^{kj} t_j$$

where $0 \leq j \leq n-1$, $\omega = e^{\frac{i2\pi}{n}}$ and $e^{i\theta} = \cos(\theta) + i\sin(\theta)$. An efficient way to evaluate the DFT is by using the FFT algorithm such that we can denote

$$\begin{aligned}
y_k &= F.t \\
y_k &= \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} \omega^{kj} t_k \\
&= \tfrac{1}{\sqrt{n}} \left[\left(\sum_{j=0}^{\frac{n}{2}-1} \omega_n^{2kj} t_{2j} \right) + \left(\sum_{j=0}^{n-1} \omega_n^{k(2j+1)} t_{2j+1} \right) \right] \\
&= \tfrac{1}{\sqrt{n}} \left[\left(\sum_{j=0}^{\frac{n}{2}-1} \omega_n^{2kj} t_{2j} \right) + \omega_n \left(\sum_{j=0}^{n-1} \omega_n^{2kj} t_{2j+1} \right) \right]
\end{aligned}$$

The matrix F is the FFT matrix. Since

$$\omega_n = e^{\frac{i2\pi}{n}}$$

it implies that

$$\omega_n^{2kj} = e^{\frac{i2\pi 2kj}{n}}$$

hence

$$\omega_{\frac{n}{2}}^{2kj} = e^{\frac{i2\pi kj}{\frac{n}{2}}}.$$

If we rewrite the expression for y_k it means that

$$y_k = \frac{1}{\sqrt{n}} \left[\left(\sum_{j=0}^{\frac{n}{2}-1} \omega_{\frac{n}{2}}^{2kj} t_{2j} \right) + \omega_n \left(\sum_{j=0}^{n-1} \omega_{\frac{n}{2}}^{2kj} t_{2j+1} \right) \right]$$

To illustrate the FFT procedure for the case when $n = 4$ where $\omega = e^{\frac{i2\pi}{4}}$ so that $\omega^4 = 1$ and $\omega^2 = -1$. Using F_n

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & 1 & \omega^2 \\ 1 & \omega^3 & \omega^2 & \omega^1 \end{pmatrix} \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{pmatrix}.$$

It is easy to see from the equation above that

$$\begin{aligned} b_0 &= t_0 + t_1 + t_2 + t_3 \\ b_1 &= t_0 + t_1\omega + t_2\omega^2 + t_3\omega^3 \\ b_2 &= t_0 + t_1\omega^2 + t_2\omega^4 + t_3\omega^6 \\ b_3 &= t_0 + t_1\omega^3 + t_2\omega^6 + t_3\omega^9 \end{aligned}$$

Since $\omega^4 = 1$ and $\omega^2 = -1$, we can express

$$\begin{aligned} b_0 &= (t_0 + t_2) + (t_1 + t_3) \\ b_2 &= \left(t_0 + t_2\omega^4\right) + \left(t_1\omega^4 + t_3\omega^6\right) = (t_0 + t_2) - (t_1 + t_3) \end{aligned}$$

Note that the solution of b_2 is easily derived from the solution b_0 since the values $(t_0 + t_2)$ and $(t_1 + t_3)$ have already been calculated. Similarly,

$$\begin{aligned} b_1 &= \left(t_0 + t_2\omega^2\right) + \left(t_1\omega + t_3\omega^3\right) = (t_0 - t_2) + \omega\,(t_1 - t_3) \\ b_3 &= \left(t_0 + t_2\omega^6\right) + \left(t_1\omega^3 + t_3\omega^9\right) = (t_0 - t_2) - \omega\,(t_1 - t_3) \end{aligned}$$

Since this procedure is completely general, one can note that the computational cost is halved in each step since we only have to find half of the unknowns. Therefore a n-dimensional matrix requires $\log(n)$ steps. It is also easy to see that each stage of the procedure has a computation cost of $O(n)$ so that the overall FFT cost is $O(n\log(n))$.

Therefore, given a $n \times n$ Toeplitz matrix with $T = \text{Toeplitz}(t)$, it implies that one can form the circulant matrix

$$C = \begin{pmatrix} T & S \\ S & T \end{pmatrix}$$

where the first column vector of the circulant matrix is defined by

$$c = \begin{pmatrix} t_0 & t_1 & \cdots & t_{n-1} & 0 & t_{1-n} & \cdots & t_{-1} \end{pmatrix}^T.$$

This means that to compute $y = Tx$, the following simple steps are undertaken:

1. Compute $\hat{c} = \text{FFT}(c)$ and $\hat{x} = \text{FFT}\begin{pmatrix} x \\ 0 \end{pmatrix}$
2. Compute $\hat{y} = \hat{c}. * \hat{x}$ where $.*$ represents component-wise multiplication
3. Compute $y = \text{IFFT}(\hat{y})$

4. Extract the first n components of y

The above FFT computation procedure is applied on all the Toeplitz matrices defined in the Gohberg-Semencul formula used to find M^{-1}.

4 Complexity and Storage Analysis

4.1 Storing S

From the computations involved in the superfast least squares Schur algorithm, there is need to store the $n \times n$ generators and the Schur complements used. Each of the four generators is of length n so that the data is stored as

$$\begin{pmatrix} a(z) & b(z) & c(z) & d(z) \\ & & S_1 & \\ & & S_2 & \end{pmatrix}.$$

We can define our recurrence relations as

$$\gamma_n = 2\gamma_{\frac{n}{2}} + bn.$$

The S_1 and S_2 stored above are equivalent structures. This uses the block Schur complement recursively defined for M_{11} and M_{22} so that

$$\begin{pmatrix} 0 & 0 \\ 0 & M_s \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & M_{22} - M_{21}M_{11}^{-1}M_{12}^T \end{pmatrix}$$

where

$$M = \begin{pmatrix} M_{11} & M_{21}^T \\ M_{21} & M_{22} \end{pmatrix}$$

Here we assume that γ_n is the length of S so that $\gamma_{\frac{n}{2}}$ is the length of S_1 and S_2. It can then be shown that the solution to the recurrence is

$$\gamma_n = n\log n + bn.$$

4.2 Computational Speed

Recall that the polynomials coefficients are stacked up in P such that

$$P(z) = \begin{pmatrix} h_{11}(z) & h_{21}(z) & h_{31}(z) & h_{41}(z) & h_{12}(z) & h_{22}(z) & \cdots & h_{44}(z) \end{pmatrix}$$

which are sixteen columns in total. Since the superfast least squares Schur algorithm calls itself twice and does convolutions with a cost of $k_1 n \log n$ then its computational cost is defined by

$$C_n = 2C_{\frac{n}{2}} + k_1 n\log n,$$

with

$$C_1 = k_0.$$

It can then be shown that this recurrence has a solution equivalent to

$$C_n = \frac{k_1}{2} n\log^2 n + \frac{k_1}{2} n\log n + k_0 n.$$

which implies that the superfast least squares Schur algorithm is of $\mathrm{O}(n\log^2 n)$.

5 Application to Image Deblurring

For discrete image processing, the convolution integral is replaced by a sum. The blurry image $B \in \Re^{n\times n}$ is obtained from the original image $X \in \Re^{n\times n}$ by the convolution process

where $0 \leq x, y, \hat{x}, \hat{y} \leq n$ and $n = 255$ so that

$$B = T_1 X T_2 + F$$

The point spread function used is seperable which implies that in our algorithms, we will choose

$$g\left(x - \hat{x}\right) = e^{-0.1(x-\hat{x})^2}$$

and

$$h\left(y - \hat{y}\right) = e^{-0.1(y-\hat{y})^2}$$

because they have zero boundary conditions. The values of s_{j-i} are defined using the point spread function $\phi\left(x - \hat{x}, y - \hat{y}\right) = g\left(x - \hat{x}\right) h\left(y - \hat{x}\right)$. Since the p.s.f is seperable, it implies that the blurred model is given by the equation

$$\begin{aligned} B_{xy} &= \sum_{\hat{x}\hat{y}} I\left(\hat{x}, \hat{y}\right) \phi\left(x - \hat{x}, y - \hat{y}\right) \\ &= \sum_{\hat{x}\hat{y}} I\left(\hat{x}, \hat{y}\right) g\left(x - \hat{x}\right) h\left(y - \hat{y}\right) \\ &= \sum_{\hat{x}} g\left(x - \hat{x}\right) \left(\sum_{\hat{y}} I\left(\hat{x}, \hat{y}\right) h\left(y - \hat{y}\right) \right) \end{aligned}$$

If we let

$$\begin{pmatrix} i_{11}^{(1)} & i_{12}^{(1)} & \cdots & i_{1n}^{(1)} \\ \vdots & \vdots & \cdots & \vdots \\ i_{(n-1)1}^{(1)} & i_{(n-1)2}^{(1)} & \cdots & i_{(n-1)(n-1)}^{(1)} \\ i_{n1}^{(1)} & i_{n2}^{(1)} & \cdots & i_{nn}^{(1)} \end{pmatrix} = \sum_{\hat{y}} I\left(\hat{x}, \hat{y}\right) h\left(y - \hat{y}\right)$$

then

$$I_1(\hat{x}, y) = \begin{pmatrix} i_{11} & i_{12} & \cdots & i_{1n} \\ \vdots & \vdots & \cdots & \vdots \\ i_{(n-1)1} & i_{(n-1)2} & \cdots & i_{(n-1)(n-1)} \\ i_{n1} & i_{n2} & \cdots & i_{nn} \end{pmatrix} \cdots \begin{pmatrix} h_0 & h_1 & \cdots & h_{n-1} \\ h_{-1} & h_0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & h_1 \\ h_{-(n-1)} & h_{-(n-2)} & \cdots & h_0 \end{pmatrix}$$

This indicates that $I_1 = I(\hat{x}, \hat{y})\, T_1$. This fact allows us to express the blurred model $B = I_2$ such that

$$\begin{aligned} I_2(x, \hat{y}) &= \sum_{\hat{x}} g(x - \hat{x})\, I_1(\hat{x}, y) \\ &= \begin{pmatrix} g_0 & g_1 & \cdots & g_{n-1} \\ g_{-1} & g_0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & g_1 \\ g_{-(n-1)} & g_{-(n-2)} & \cdots & g_0 \end{pmatrix} \cdots \begin{pmatrix} i^{(1)}_{11} & i^{(1)}_{12} & \cdots & i^{(1)}_{1n} \\ \vdots & \vdots & \cdots & \vdots \\ i^{(1)}_{(n-1)1} & i^{(1)}_{(n-1)2} & \cdots & i^{(1)}_{(n-1)(n-1)} \\ i^{(1)}_{n1} & i^{(1)}_{n2} & \cdots & i^{(1)}_{nn} \end{pmatrix} \\ &= T_2 \quad I(\hat{x}, \hat{y})\, T_1 \end{aligned}$$

As a result it is clear from the results above that the matrices $T_1$and T_2 are defined using the p.s.f to yield

$$\begin{aligned} T_1 &= T_2 \\ &= \begin{pmatrix} s_0 & s_{-1} & s_{-2} & s_{-3} & \cdots & s_{-(n-1)} \\ s_1 & s_0 & s_{-1} & s_{-2} & \ddots & s_{-(n-2)} \\ \vdots & s_1 & s_0 & s_{-1} & \ddots & \vdots \\ \vdots & \ddots & s_1 & s_0 & \ddots & \vdots \\ s_{n-2} & \ddots & \ddots & \ddots & \ddots & s_{-1} \\ s_{n-1} & s_{n-2} & \cdots & \cdots & s_1 & s_0 \end{pmatrix} \end{aligned} \tag{5}$$

so that $s_{ij} = s_{j-1}$. These matrices are used to model convolution. Since both T_1 and T_2 are square and symmetric Toeplitz matrices, then $T_1 \in \Re^{n\times n}$ and $T_2 \in \Re^{n\times n}$.

Recall also that the noise level $F \in \Re^{n\times n}$ induced into the system implied that $B = T_1XT_2 + F$. In our experiments, the noise level is defined using the randomized function with the standard deviation of the error level being *std*. This parameter can be varied along with the regularization parameter α. Theoretically and experimentally, one can determine an optimal regularization parameter using either the generalized cross-validation [11], [12] or the $L-$curve [11], [12]. In these articles, the L-curve is defined as a plot of the squared estimate norm $\|X\|_2^2$ against the residual sum of squares

$$\left\| \begin{pmatrix} T_1 \\ \alpha I \end{pmatrix} Y - \begin{pmatrix} B \\ 0 \end{pmatrix} \right\|_F^2$$

where $Y = XT_2$. It is suggested that an optimal trade off between the estimate norm and

the RSS is found at the corner of the L-Curve. The variance squared is defined as

$$\frac{RSS}{n} = \frac{\| \begin{pmatrix} T_1 \\ \alpha I \end{pmatrix} Y - \begin{pmatrix} B \\ 0 \end{pmatrix} \|_F^2}{n} = s^2$$

More information about the generalized cross-validation procedure is detailed in the papers. In our experiments, α and the *std* values are varied by trial and error to determine their optimal values.

In defining the error in the image, we use the MATLAB statement

$$F = std * \text{randn}(n, n),$$

where the RANDN function produces normally distributed random numbers such that randn(n, n) is an $n \times n$ matrix with random entries chosen from a normal distribution. This yields a zero mean value, a variance of one and a variable standard deviation.

Therefore if the matrix under consideration is $M = T_1^T T_1 + \alpha^2 I$, one can easily create the generator matrix $\hat{G}$ from which we extract the column vectors of $\hat{G}$ to be

$$\begin{aligned} \hat{G} &= \begin{pmatrix} s_0 & \sqrt{\|t_{-1}\|_2^2 + \alpha^2} & 0 & 0 \\ t_1 & \frac{T_2^T t_{-1}}{\sqrt{\|t_{-1}\|_2^2+\alpha^2}} & \hat{t}_{-1} & \frac{T_2^T t_{-1}}{\sqrt{\|t_{-1}\|_2^2+\alpha^2}} \end{pmatrix} \\ &= \begin{pmatrix} a_0 & b_0 & c_0 & d_0 \\ a_1 & b_1 & c_1 & d_1 \\ \vdots & \vdots & \vdots & \vdots \\ a_{n-2} & b_{n-2} & c_{n-2} & d_{n-2} \\ a_{n-1} & b_{n-1} & c_{n-1} & d_{n-1} \end{pmatrix}. \end{aligned}$$

From this definition of $\hat{G}$, the regularization parameter α indicated is varied along with the standard deviation in our superfast linear least squares Schur algorithm as we seek to stabilize

our solution and search for the Tikhonov solution. As indicated, the generator matrix can be expressed in the form of polynomials. This can be seen in detail in the expression for the generators of the matrix M calculated using the Gohberg-Semencul formular. Note that these polynomials can be multiplied out with some computational speedup via the FFT.

Having defined the generators as the column vectors and expressed them as polynomials, one can then invoke the sflsschur algorithm. The algorithm performs the deblurring routine which is done via FFT as seen in the invoking of the function fconv. The process is done row-wise and column-wise to produce the deblurred image. The need to perform row-wise and column-wise deblurring is attributed to the fact that our image distortions involve a seperable p.s.f.

Once the deblurred image is obtained it becomes necessary to evaluate our image reconstruction results. In image restoration, the need to measure the quality of the restored image is essential. There are two commonly used measures of measuring image quality: the Mean-Squared Error(MSE) , Peak Signal-to-Noise Ratio(PSNR) and the relative error. The MSE is defined as

$$E_{MSE} = \frac{1}{n^2} * \sum_{k=0}^{n-1} \sum_{i=0}^{n-1} \left(\hat{B}[k,i] - X[k,i] \right)^2$$

where $\hat{B}[n,n]$ is the restored image while $X[n,n]$is the original image. On the other hand, the Peak Signal-to-Noise Ratio($PSNR$) is often used in engineering to measure the quality of the restored images and is defined as

$$PSNR = -10\log\frac{E_{MSE}}{S^2}$$

where S is the maximum pixel value. The $PSNR$ is measured in decibels (dB). It should be noted that $PSNR$ is a good measure for comparing restoration results for the same image, but when comparing different images, $PSNR$ is rendered meaningless. In both measurements, a lower value implies a better restored image and the same argument if otherwise. As numerical analysts, it is customary to use the relative error as a measure of

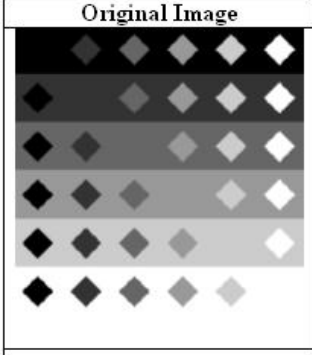
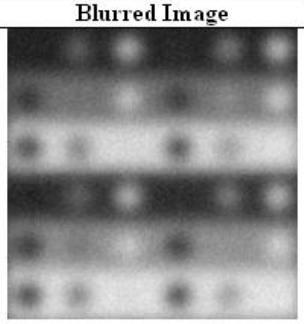
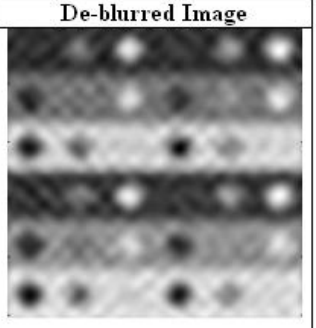
Original Image	Blurred Image	De-blurred Image
		
α	2	
PSF	$e^{\frac{-x^2}{10}}$, $x \in [0,255]$	
Stdev.	0.05	
MSE	0.0074	

Figure 1: Deblurring with $\alpha = 2$

deviation from the actual results. In this research, we calculate the relative error to be

$$RelError = \frac{max\,(B[i,j] - X[i,j])}{max\,(X[i,j])}.$$

In our experiments we will evaluate the relative errors as a measure of the quality of the restored images as opposed to using $PSNR$ or E_{MSE}.

6 Results and Analysis

As seen from the image outputs, by varying the regularization parameter α along with the standard deviation *std* in our tests, one can examine the blurring effect on the images. A good choice of the point spread function (PSF) is also needed.

From Figure 5, it is evident that as α and the standard deviation is decreased, the relative error decreases respectively. This implies that there is an optimal value of α and the standard deviation near zero for which one can recreate sharp images. It is suggested that by plotting the L-curve these optimal values can be determined apriori. A good choice of α and the *std* plays a greater role in the deblurring process since it is these parameters that control the

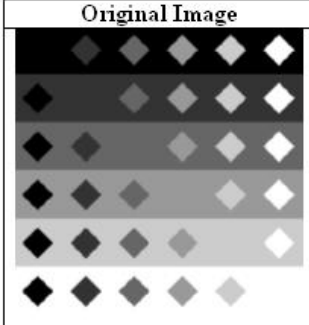
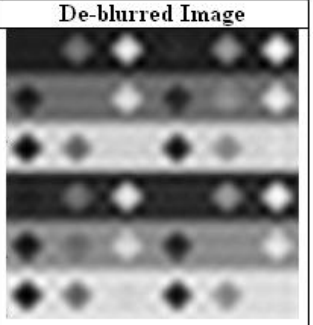
Original Image	Blurred Image	De-blurred Image
		
α	0.5	
PSF	$e^{\frac{-x^2}{10}}$, $x \in [0,255]$	
Stdev.	0.0006	
MSE	0.0069	

Figure 2: Deblurring with $\alpha = 0.5$

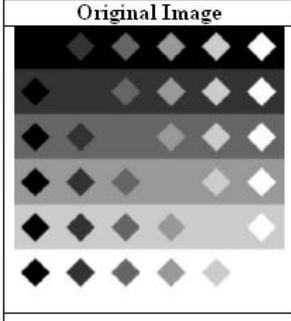
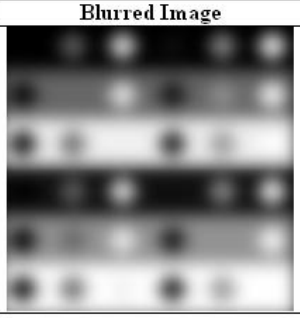
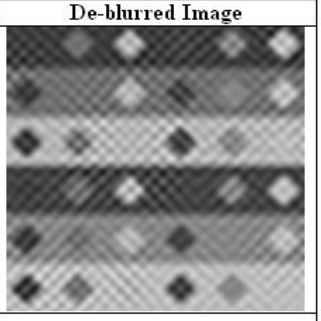
Original Image	Blurred Image	De-blurred Image
		
α	0.05	
PSF	$f(x) = e^{\frac{-x^2}{10}}$, $x \in [0,255]$	
Standard Deviation	0.0001	
Relative Error	0.0084	

Figure 3: Deblurring with $\alpha = 0.05$

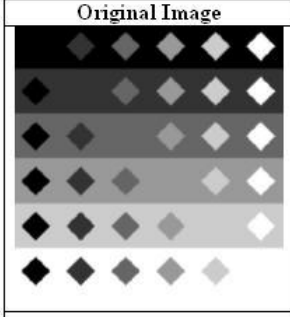
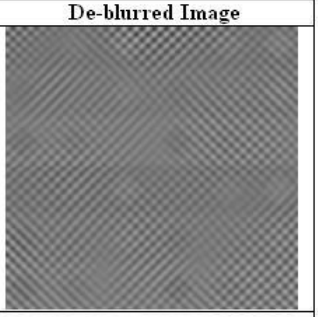
Original Image	Blurred Image	De-blurred Image
		
α	0.001	
PSF	$f(x) = e^{\frac{-x^2}{10}}, \; x \in [0,255]$	
Standard Deviation	0.000001	
Relative Error	0.1538	

Figure 4: Deblurring with $\alpha = 0.001$

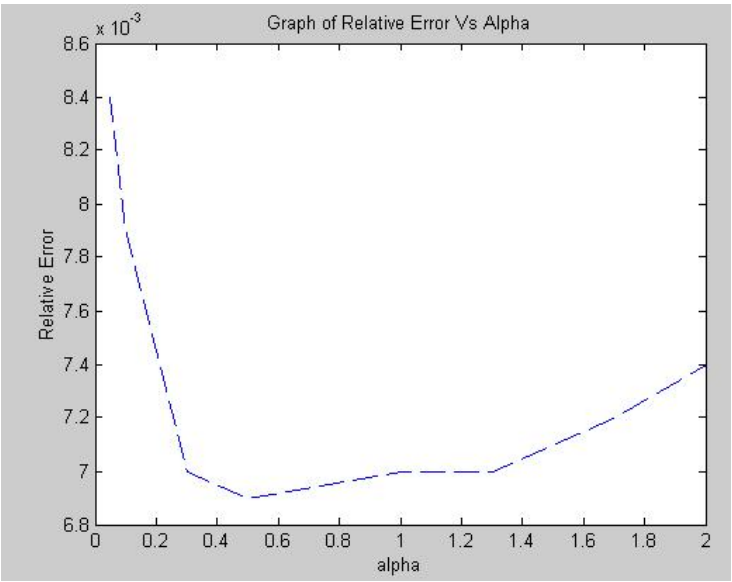


Figure 5: Relative Error vs. Alpha

Table 1: Errors for large α

α	3	2.5	2	1.7	1.3	1
Standard Deviation	1	0.1	0.05	0.01	0.005	0.001
Relative Error	0.0209	0.0177	0.0074	0.0072	0.0070	0.0070

Table 2: Errors for small α

α	0.5	0.1	0.05	0.0125	0.001
Standard Deviation	0.0069	0.0002	0.0001	0.000025	0.000001
Relative Error	0.0070	0.0079	0.0084	0.199	0.1538

degree of image details that are reconstructed. In this experiment, the point spread function used is seperable which implies that in our algorithms, we will choose

$$g(x) = e^{-0.1\left(x^2\right)}$$

and

$$h(y) = e^{-0.1\left(y^2\right)}$$

because they have zero boundary conditions leading to the construction of the Toeplitz matrix (5). When the algorithm is run on varied values of α and standard deviation, the images above were attained. Note from the results obtained that when lower α and standard deviation values is used, there is a certain threshhold when the images produced are clearly sharp beyond which they again begin to appear blurred. This can be seen in the tabulated results in Table 1 and Table 2.

Plotting a graph of the relative errors against α indicates that for large values of the regularization parameter, the relative errors are magnified. Equivalently, if we reduce the regularization parameter significantly closer to zero, the errors become magnified as well. In both cases, the reconstructed image becomes even more blurred. However, it is evident from our results that the standard deviation and regularization values closer to zero yield better results leading to good image reconstruction results. This is in contrast to the case when

we use larger standard deviation and regularization values which yield even more blurred images. The case when $\alpha = 0$ is equivalent to the ill-conditioned scenario which produces the "naive" solution. From the reconstruction results it is easy to see that the naive solution corresponds to the situation when no reconstructed image output is achieved at all.

7 Conclusion

It is clear from this research that the Tikhonov regularization technique allows one to stamp our errors when solving linear least squares problems. It is also a viable method for solving large, structured linear least squares problems. Its results has been shown to be applicable in image deblurring. In this thesis, we have in effect shown that the regularization parameter α and the standard deviation controls the quality of the reconstructed image. It is also shown conclusively that the choice of α and the standard deviation when educatedly chosen yields better image reconstruction results. This is shown when we pick the values of α and the standard deviation that are significantly small leading to sharper reconstructed images. The implication is that there exists optimal values for which α and the standard deviation can produce sharp images.

From the graph of relative error against α, we can infer that lower values of α yield lower relative errors. One other greater result in our experiments is the speed of doing the computations. We have shown how we can take advantage of the FFT process in our algorithms and this allows us to work with large structured Toeplitz matrices easily without which our computations would be formidable. More importantly, we have shown conclusively that Tikhonov regularization can be used to damp out errors in ill-conditioned problems with the use of the regularization parameter α. Even more clearly, it is seen that this parameter can be used to control the level of image deblurring the field of image restoration.

References

[1] Michael Stewart and Paul Van Dooren, Stability Issues in the Factorization of Structured Matrices, SIAM J. Matrix Anal. Appl. Volume 18 Number 1, p. 104-118, Jan 1997.

[2] P. C. Hansen, De-Convolution and Regularization with Toeplitz Systems, Kluwer Academic Press, Lyngby, Denmark, 2002.

[3] G. H. Golub and C. Van Loan, Matrix Computations, Second edition, Johns Hopkins Press, Baltimore, Maryland, 1989.

[4] Michael Stewart, A Superfast Toeplitz Solver with Improved Numerical Stability, J. Matrix Anal. Appl. Volume 25, Number 3 pp. 669-693, 2003

[5] http://www.siam.org/siamnews/general/images.htm

[6] Sondhi, Benesty etal, An Augumented Iterative Method for Large Linear Toeplitz Systems, Bell Labs - Lucent Technologies, Murray Hill, NJ, pp.1-3. 1999

[7] D. L. Phillips, A technique for the numerical solution of certain integral equations of the first kind. J. Assoc. Computing. March. 9(1962), 84-97, MR 24:B534

[8] Tikhonov A. N, Solution of incorrectly formulated problems and the regularization method. Soviet Math Dokl 4, 1035-1038. English translation of Dokl Akad Nauk SSSR 151, 1963, 501-504

[9] http://homepages.inf.ed.ac.uk/rbf/HIPR2/noise.htm

[10] http://www.icaen.uiowa.edu/~dip/LECTURE/PreProcessing4.html

[11] Hansen, P. C., 1998: Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion. SIAM Monogr. on Mathematical Modeling and Computation, Society for Industrial and Applied Mathematics.

[12] Wahba, G., 1990: Spline Models for Observational Data. CBMS-NSF Regional Conference Series in Applied Mathematics, Vol. 59, Society for Industrial and Applied Mathematics.