ADVANCES IN SLIDING WINDOW SUBSPACE TRACKING

BY

TIMOTHY M. TOOLAN

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

ELECTRICAL ENGINEERING

UNIVERSITY OF RHODE ISLAND

2005

DOCTOR OF PHILOSOPHY DISSERTATION

OF

TIMOTHY M. TOOLAN

APPROVED:

Dissertation Committee:

Major Professor Donald Tufts

James Cooley

Richard Vaccaro

Lewis Pakula

UNIVERSITY OF RHODE ISLAND

2005

ABSTRACT

This dissertation is concerned with the task of efficiently and accurately tracking the singular values and left singular vectors of a rapidly changing dominant column space of a matrix, in which a column of the matrix is replaced each time step. As part of this task, the dimension of this dominant subspace is determined automatically for each time step.

Two methods for determining the singular values and left singular vectors of this dominant columnspace are presented. The first method, which is an exact method, will update all of the singular values and left singular vectors using the rank-two secular function. The derivation of this function, and its properties, are original contributions of this dissertation. This method requires a single $O(n^3)$ matrix product to rotate the singular vectors using direct multiplication, and all other computation is $O(n^2)$. The second method, the Improved Fast Adaptive Subspace Tracking (IFAST) method, will give accurate approximations of the r largest singular values and corresponding left singular vectors in $O(nr^2)$ time. An accuracy analysis of the approximation error of the second method, using the rank-two secular function from the first method, is presented.

The block Hankel matrix structure is presented, which can give improved SNR for exponential signals in sensor arrays, at the expense of beam-width.

The two rank determination methods of Shah and Tufts, where one is for time-series (Hankel) data and the other is for unstructured sensor array data, are combined into a single general method which works with unstructured, Hankel, and block Hankel matrices. An efficient way to calculate the thresholds used by this method is presented, allowing use in real-time applications.

ACKNOWLEDGMENTS

I would like to thank the members of my thesis committee. Thank to Dr. Donald Tufts for his guidance and patience over the years. His ability to ask the right questions greatly improved the quality of this thesis. Thanks to Dr. James Cooley for taking time out of retirement to meet with me about my work. His ability to instantly grasp ideas that I have been working on for a long time, then give useful feedback has been very helpful. Thanks to Dr. Richard Vaccaro for being a good boss and a good committee member. His knowledge of the singular value decomposition, and matrix operations in general has helped a lot. Thank to Dr. Lewis Pakula for giving feedback, even when he felt that the work was not necessarily in his field. Thanks to Dr. James Baglama for his help with the work on the rank-two secular equation, even though he is not on the core committee. Thanks to Dr. G. Faye Boudreaux-Bartels for taking the time to be on the committee, even though she is busy with her new job as department chair (and thus my new boss). Thanks to Dr. Orlando Merino for being the outside member at the last minute.

Thanks to the people who worked with me in the department, for picking up the slack when I was busy working on my research. Thanks to Ashwin Sarma for some useful and enjoyable discussions.

Thanks to Dr. Norman Owsley for the use of his simulated sonar array data, which is used in the results section of manuscript 4. Thanks to Don Knuth for TeX, and Leslie Lamport for LaTeX, without which, writing this dissertation would be much more difficult.

Finally, I would like to thank both my wife and my parents for supporting and encouraging me over the years.

To Su

PREFACE

This dissertation is written in manuscript form, and investigates various issues related to subspace tracking.

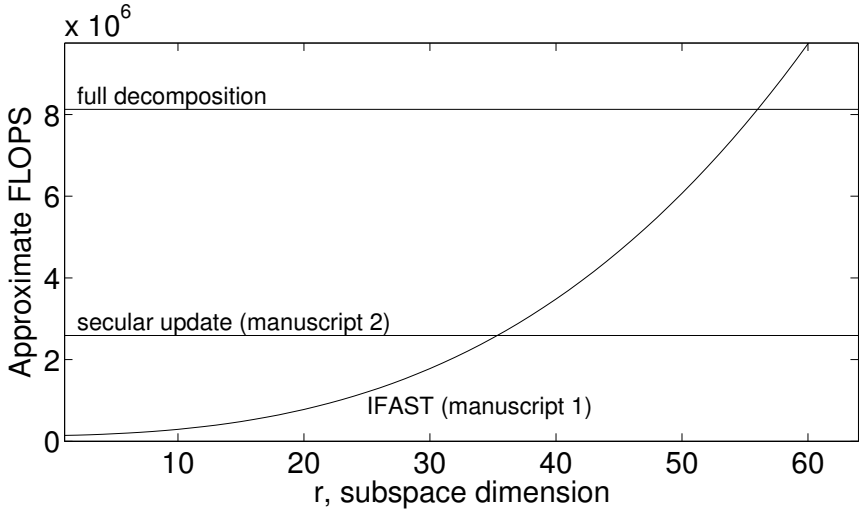
Manuscript 1 introduces the IFAST algorithm, which is an accurate $O(nr^2)$ subspace tracking algorithm, for updating the r largest singular values and left singular vectors when a column is replaced in a matrix. The IFAST-S algorithm is presented, which allows the algorithm to start without an initial SVD. A comparative analysis of the full dimension secular equation from manuscript 2 is presented, and shows why the IFAST approximation is accurate. This work is an extension of the work previously published as

> T. M. Toolan and D. W. Tufts, "Improved fast adaptive subspace tracking," in *Proceedings Thirteenth Adaptive Sensor Array Processing Workshop (ASAP05)*, MIT Lincoln Laboratory, Lexington MA, June 2005.

Manuscript 2 introduces a method for calculating the eigendecomposition of the sum of an $n \times n$ diagonal matrix and two rank-one matrices of the form $\tilde{G} = D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H$, in $O(n^2)$ time. This allows one to update the singular values and left singular vectors of a matrix where one column has been replaced, with a single $O(n^3)$ matrix product. The rank-two secular equation is presented, whose roots are the eigenvalues of a matrix of the form of $\tilde{G}$, as well as a non-iterative method to determine the eigenvectors of $\tilde{G}$. All non-numerical pathological cases are addressed, which not only are the basis for a practical algorithm, but also give insight into what is really happening. The rank-two secular equation is used to analyze the accuracy of the IFAST algorithm in manuscript 1. This work has been submitted for publication.

The following figure shows the computational savings using the IFAST algorithm from manuscript 1, and the secular update method from manuscript 2,

vs. calculating the full singular value decomposition of a 64×64 complex matrix where one column has changed.



Manuscript 3 introduces a method for reducing computation of the FAST algorithm by Real, Tufts, and Cooley when the data matrix is Hankel. This method is used as the basis for the IFAST-F algorithm in manuscript 1. This work was previously published as

> J. W. Cooley, T. M. Toolan, and D. W. Tufts, "A subspace tracking algorithm using the Fast Fourier Transform," *IEEE Signal Processing Letters*, vol. 11, no. 1, Jan. 2004.

Manuscript 4 introduces the *block Hankel* matrix structure, which can give improved SNR for sensor arrays at the expense of beam-width. This manuscript also combines the two rank determination methods of Shah and Tufts into a single general method which works with unstructured, Hankel, and block Hankel matrices. This rank determination method is based on a threshold test using sums of singular values, and a method to efficiently calculate the thresholds for use in a real-time implementation is presented. The key elements of this efficient calculation consist of using a scaled Chi-Square distribution for a Chi-Square mixture, and using reduced dimension, noise only, matrices to determine higher rank thresholds. The rank determination method presented here is used to determine the signal subspace

dimension in manuscript 1. This work was previously published as

> T. M. Toolan and D. W. Tufts, "Detection and estimation in non-stationary environments," in *Proc. IEEE Asilomar Conference on Signals, Systems & Computers*, Nov. 2003, pp. 797–801.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

MANUSCRIPT 1

The Improved Fast Adaptive Subspace Tracking (IFAST) Algorithm

1.1 Introduction

A new subspace tracking algorithm which gives accurate estimates of the r largest singular values and corresponding left singular vectors of overlapping rectangular matrices is presented. This $O(nr^2)$ algorithm, which we will call the *Improved Fast Approximate Subspace Tracking* (IFAST) algorithm, has evolved from the *Fast Approximate Subspace Tracking* (FAST) algorithm by Real, Tufts, and Cooley [1], but has significantly better accuracy and computational efficiency. Additionally, we present techniques for starting without an initial singular value decomposition (SVD), advancing the rectangular data window more than one column per iteration, and taking advantage of Hankel structure to reduce computation.

A detailed analytical analysis of the effect that advancing a rectangular data window by one column has on its full SVD vs. the effect it has on the approximate SVD which is produced by the IFAST algorithm is presented. This analysis is based on the rank-two secular equation from manuscript 2, which is closely related to the characteristic polynomial.

We have been motivated by problems of detection and estimation in a non-stationary environment. Often the "signal" subspace is really a rapidly varying subspace of interference or clutter, and we wish to track the subspace in order to facilitate removal of the interference. Two examples are the rapidly time-and-space varying clutter in multi-spectral images [2], and Terrain Scattered Interference in airborne radar [3].

We start with an $n \times c$ matrix M, whose columns consist of sequential complex-valued samples in time and/or space. For example, each column may be the output

of an array of sensors at a given time, or a new column in a Hankel matrix created from a single sensor. M is of the form

$$M = S + N \tag{1}$$

where S is a rank r signal matrix and N is a full rank noise matrix. Our goal is to efficiently and accurately determine the signal subspace dimension, r, along with the r largest singular values and corresponding left singular vectors of M. The problem of determining r when we have the singular values of M has been addressed in [4] and [5], and generalized in manuscript 4 and [6].

1.2 The IFAST Algorithm

In this section we present the *improved fast adaptive subspace tracking* (IFAST) algorithm, and give some examples of its performance.

1.2.1 The Goal of IFAST

Given a sequence of $c+1$, length n, column vectors, $\boldsymbol{x}_{t-c} \cdots \boldsymbol{x}_t$, we can define the two $n \times c$ matrices

$$M = \left[\begin{array}{ccccc} \boldsymbol{x}_{t-c} & \boldsymbol{x}_{t-c+1} & \boldsymbol{x}_{t-c+2} & \cdots & \boldsymbol{x}_{t-1} \end{array} \right], \tag{2}$$

$$\tilde{M} = \left[\begin{array}{ccccc} \boldsymbol{x}_{t-c+1} & \boldsymbol{x}_{t-c+2} & \cdots & \boldsymbol{x}_{t-1} & \boldsymbol{x}_t \end{array} \right]. \tag{3}$$

The matrix $\tilde{M}$ can be created from the matrix M by discarding its leftmost column, then appending the vector $\boldsymbol{x}_t$ to the right, thus the matrices M and $\tilde{M}$ will have all but one column in common. We can write the SVD of M as

$$M = U \Sigma V^H = \left[\begin{array}{cc} U' & U^\perp \end{array} \right] \left[\begin{array}{cc} \Sigma' & 0 \\ 0 & \Sigma^\perp \end{array} \right] \left[\begin{array}{cc} V' & V^\perp \end{array} \right]^H, \tag{4}$$

where $\Sigma' \in \mathbb{R}^{r \times r}$ are the r largest singular values of M, and $U' \in \mathbb{C}^{n \times r}$ are the corresponding left singular vectors. The superscript H represents conjugate transpose of a matrix.

Assuming we have the r largest singular values and corresponding left singular vectors of M, we would like to determine the r (or possibly $r+1$) largest singular values and corresponding left singular vectors of $\tilde{M}$, without the computation required to determine a full SVD of $\tilde{M}$.

1.2.2 The IFAST Approximation

Because M shares all but one column with $\tilde{M}$, it is reasonable to assume that the column space spanned by the r largest left singular vectors of M, must be a reasonable approximation to the column space spanned by the r largest left singular vectors of $\tilde{M}$, except possibly for the contribution from the column we are adding and the column we are discarding. To include the contribution from those two columns to the column space of U', we can augment U' by the column space of $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t-c}$ that is orthogonal to U' to get an $r+2$ dimensional subspace. Partial augmentation of U' has been used previously in both [1] and [7].

In IFAST, we first use a Gram-Schmidt method to create the matrix $Q \in \mathbb{C}^{n\times 2}$, which is an orthonormal basis for the column space defined by the vectors that we are adding and discarding, that is orthogonal to U'. We can write the two columns of $Q = [\, \boldsymbol{q}_1 \,|\, \boldsymbol{q}_2]$ as

$$\begin{aligned} \boldsymbol{z}_1 &= \left(I - U'U'^H\right)\boldsymbol{x}_t, & \boldsymbol{q}_1 &= \boldsymbol{z}_1/\|\boldsymbol{z}_1\|_2, \\ \boldsymbol{z}_2 &= \left(I - [\, U' \,|\, \boldsymbol{q}_1 \,]\,[\, U' \,|\, \boldsymbol{q}_1 \,]^H\right)\boldsymbol{x}_{t-c}, & \boldsymbol{q}_2 &= \boldsymbol{z}_2/\|\boldsymbol{z}_2\|_2. \end{aligned} \tag{5}$$

We next define the rank $r+2$ matrix, $\tilde{M}'$, to be the projection of $\tilde{M}$ into the column space defined by $[\, U' \,|\, Q \,]$, which we will write as

$$\tilde{M}' = [\, U' \,|\, Q \,][\, U' \,|\, Q \,]^H \tilde{M}. \tag{6}$$

The essence of IFAST, is that we assume the singular values and left singular vectors of $\tilde{M}'$ are good approximations to the $r+2$ largest singular values and

corresponding left singular vectors of $\tilde{M}$, therefore we substitute the SVD of $\tilde{M}$ with the SVD of $\tilde{M}'$. This is the only approximation that takes place in the algorithm, and is reasonable because it is motivated by the fact that we retain a Frobenius norm approximation to the signal subspace which is at least as good as that from our previous iteration [1]. In later sections, we will analyze this approximation in detail.

1.2.3 Basic Theory

Now that we have decided that we are going to use the SVD of

$$\tilde{M}' = \tilde{U}'\tilde{\Sigma}'\tilde{V}'^H, \tag{7}$$

as our approximation to the SVD of $\tilde{M}$, we would like to compute it as efficiently as possible.

The following theorem will show how to replace the SVD of the $n \times c$ matrix $\tilde{M}'$ with an equivalent problem that is the eigendecomposition of the much smaller $(r+2) \times (r+2)$ matrix $\tilde{F}$, where

$$\tilde{F} = [\, U' \,|\, Q \,]^H \tilde{M}\tilde{M}^H [\, U' \,|\, Q \,] = \tilde{U}_f \tilde{\Sigma}_f \tilde{U}_f^H. \tag{8}$$

It will be shown that the singular values and left singular vectors of $\tilde{M}'$ will be

$$\tilde{\Sigma}' = \sqrt{\tilde{\Sigma}_f} \qquad \text{and} \qquad \tilde{U}' = [\, U' \,|\, Q \,]\tilde{U}_f. \tag{9}$$

Theorem 1. *Let $M \in \mathbb{C}^{n \times c}$ be any matrix, and let $E' \in \mathbb{C}^{n \times k}$ have k orthonormal columns, with $k \leq n$. Let M' be the projection of M onto the column space of E',*

$$M' = E'E'^H M, \tag{10}$$

and let the SVD of $M' = U'\Sigma'V'^H$. If we define the matrix $F \in \mathbb{C}^{k \times k}$, along with its eigendecomposition, as

$$F = E'^H M M^H E' = U_f \Sigma_f U_f^H, \tag{11}$$

then we can write the non-zero singular values, and corresponding left singular vectors of M' as

$$\Sigma' = \sqrt{\Sigma_f}, \qquad U' = E'U_f. \tag{12}$$

Proof. If we multiply (11) by $U_f^H E'^H E'$ from the left, and $E'^H E' U_f$ from the right, (remember $E'^H E' = I$, and U_f is unitary), we get

$$U_f^H E'^H E' E'^H M M^H E' E'^H E' U_f = \Sigma_f. \tag{13}$$

Substituting $E'E'^H M = M'$, we get

$$U_f^H E'^H M' M'^H E' U_f = \Sigma_f. \tag{14}$$

If we augment $E'U_f$ with $U^\perp$ from (4) to create the unitary matrix $[\, E'U_f \mid U^\perp \,] \in \mathbb{C}^{n \times n}$, then because the columns of $U^\perp$ must come from the null space of $E'U_f$, which is $(I - E'E'^H)$, they will be orthogonal to M'. This allows us to write

$$\left[\ E'U_f \,\middle|\, U^\perp\ \right]^H M' M'^H \left[\ E'U_f \,\middle|\, U^\perp\ \right] = \begin{bmatrix} \Sigma_f & 0 \\ 0 & 0 \end{bmatrix}, \tag{15}$$

which from Theorem 2.5.2 in [8], is the SVD of $M'M'^H$. We can write the SVD of $M'M'^H$ as $U'\Sigma'V'^H V'\Sigma'^H U'^H = U'\Sigma'^2 U'^H$, thus $\Sigma_f = \Sigma'^2$ and $E'U_f = U'$. ■

The important point of Theorem 1 is that it applies for any M and E', in which the columns of E' are mutually orthonormal. If E' were left singular vectors of M, Theorem 1 would be obvious, but this will not be the case in the IFAST algorithm, because U' will contain approximations to the real singular vectors, therefore we need Theorem 1.

Now that we have reduced the SVD of $\tilde{M}'$ from (6) to the eigendecomposition of $\tilde{F}$ from (8), we would further like to reduce computation. It can be seen from (8), that creating the matrix $\tilde{F}$ requires the matrix product of an $(r+2) \times n$ matrix with an $n \times c$ matrix, plus the matrix product of an $(r+2) \times c$ matrix with

a $c \times (r+2)$ matrix. These $O(ncr)$ and $O(cr^2)$ matrix products will dominate the computation over the $O(r^3)$ SVD of $\tilde{F}$ for most values of r.

We can write $\tilde{F}$ from (8) as the block matrix

$$\tilde{F} = \left[\begin{array}{c|c} U'^H \tilde{M}\tilde{M}^H U' & U'^H \tilde{M}\tilde{M}^H Q \\ \hline Q^H \tilde{M}\tilde{M}^H U' & Q^H \tilde{M}\tilde{M}^H Q \end{array}\right]. \tag{16}$$

The following theorem will show how to replace the upper $r \times r$ block of (16), $U'^H \tilde{M}\tilde{M}^H U'$, with the matrix

$$\tilde{D} = \Sigma'^2 - U'^H \boldsymbol{x}_{t-c}\boldsymbol{x}_{t-c}^H U' + U'^H \boldsymbol{x}_t \boldsymbol{x}_t^H U', \tag{17}$$

which is the sum of a diagonal matrix and two rank-one matrices, and can be computed in $O(nr)$. This allows us to write $\tilde{F}$ as

$$\tilde{F} = \left[\begin{array}{c|c} \tilde{D} & U'^H \tilde{M}\tilde{M}^H Q \\ \hline Q^H \tilde{M}\tilde{M}^H U' & Q^H \tilde{M}\tilde{M}^H Q \end{array}\right]. \tag{18}$$

Theorem 2. *Let $M \in \mathbb{C}^{n\times c}$ be any matrix, and let $E' \in \mathbb{C}^{n\times k}$ have k orthonormal columns, with $k \leq n$. Let M' be the projection of M onto the column space of E',*

$$M' = E'E'^H M,$$

and let the SVD of $M' = U'\Sigma'V'^H$. Let $\tilde{M} \in \mathbb{C}^{n\times c}$ be created from M by discarding its leftmost column, then appending the vector $\boldsymbol{x}_t$ to the right, thus

$$\begin{aligned} M &= \left[\begin{array}{cccccc} \boldsymbol{x}_{t-c} & \boldsymbol{x}_{t-c+1} & \boldsymbol{x}_{t-c+2} & \cdots & \boldsymbol{x}_{t-1} \end{array}\right], \\ \tilde{M} &= \left[\begin{array}{cccccc} \boldsymbol{x}_{t-c+1} & \boldsymbol{x}_{t-c+2} & \cdots & \boldsymbol{x}_{t-1} & \boldsymbol{x}_t \end{array}\right]. \end{aligned}$$

If we define

$$\tilde{D} = U'^H \tilde{M}\tilde{M}^H U', \tag{19}$$

where U' are the left singular vectors of M', then $\tilde{D}$ can be written as

$$\tilde{D} = \Sigma'^2 - U'^H \boldsymbol{x}_{t-c}\boldsymbol{x}_{t-c}^H U' + U'^H \boldsymbol{x}_t \boldsymbol{x}_t^H U', \tag{20}$$

which is the sum of the diagonal matrix, Σ'^2, and two rank-one matrices.

Proof. We can write MM^H and $\tilde{M}\tilde{M}^H$ as the sum of c outer products [8],

$$MM^H = \sum_{i=t-c}^{t-1} \boldsymbol{x}_i \boldsymbol{x}_i^H, \qquad \tilde{M}\tilde{M}^H = \sum_{i=t-c+1}^{t} \boldsymbol{x}_i \boldsymbol{x}_i^H, \tag{21}$$

thus we can write

$$\tilde{M}\tilde{M}^H = MM^H - \boldsymbol{x}_{t-c}\boldsymbol{x}_{t-c}^H + \boldsymbol{x}_t \boldsymbol{x}_t^H. \tag{22}$$

Substituting (22) into (19), we get

$$\tilde{D} = U'^H MM^H U' - U'^H \boldsymbol{x}_{t-c}\boldsymbol{x}_{t-c}^H U' + U'^H \boldsymbol{x}_t \boldsymbol{x}_t^H U'. \tag{23}$$

From (12), we can write $E' = U' U_f^H$, which combined with (11) gives us

$$U_f U'^H MM^H U' U_f^H = U_f \Sigma_f U_f^H. \tag{24}$$

Multiplying (24) by U_f^H from the left, and U_f from the right, and substituting $\Sigma_f = \Sigma'^2$ from (12), we get $U'^H MM^H U' = \Sigma'^2$, which is a diagonal matrix. ■

The important point of Theorem 2 is that it says that the matrix U' will diagonalize MM^H, even though the columns of U' may or may not be singular vectors of MM^H. To illustrate this point, assume the columns of U' are not singular vectors of MM^H. Theorem 2 says $U'^H(MM^H)U' = \Sigma'^2$, and this implies both $\boldsymbol{u}_i^H(MM^H)\boldsymbol{u}_i = \sigma_i^2$ and $\boldsymbol{u}_i^H(MM^H)\boldsymbol{u}_j = 0$ for $i, j = 1, 2, \ldots, r$, $i \neq j$, must be satisfied. What is not satisfied is the eigenequation, thus $(MM^H)\boldsymbol{u}_i \neq \sigma_i^2 \boldsymbol{u}_i$.

1.2.4 The Steps of the Algorithm

In this section, we give the steps of the IFAST algorithm, which will calculate the singular values and left singular vectors of $\tilde{M}'$ from (6). The steps of the algorithm are listed in Table 1. At the beginning of the algorithm, we have our previous and current data matrices, M and $\tilde{M}$, along with the r largest singular values and corresponding left singular vectors of M which were determined in the

Table 1. Steps of the IFAST Algorithm

Step		Description
1)	$\boldsymbol{z}_1 = \left(I - U'U'^H\right)\boldsymbol{x}_t$ $\boldsymbol{q}_1 = \boldsymbol{z}_1 / \left\|\boldsymbol{z}_1\right\|_2$ $\boldsymbol{z}_2 = \left(I - [\, U' \mid \boldsymbol{q}_1 \,]\,[\, U' \mid \boldsymbol{q}_1 \,]^H\right)\boldsymbol{x}_{t-c}$ $\boldsymbol{q}_2 = \boldsymbol{z}_2 / \left\|\boldsymbol{z}_2\right\|_2$ $Q = [\, \boldsymbol{q}_1 \mid \boldsymbol{q}_2 \,]$	Use a Gram-Schmidt method to create the matrix $Q \in \mathbb{C}^{n\times 2}$, which is an orthonormal basis for the column space defined by the vectors that we are adding and discarding from $\tilde{M}$, that is orthogonal to U'. See eqn. (5).
2)	$\tilde{D} = \Sigma'^2 - U'^H \boldsymbol{x}_{t-c}\boldsymbol{x}_{t-c}^H U' + U'^H \boldsymbol{x}_t \boldsymbol{x}_t^H U'$ $\tilde{F} = \left[\begin{array}{c\|c} \tilde{D} & U'^H \tilde{M}\tilde{M}^H Q \\ \hline Q^H \tilde{M}\tilde{M}^H U' & Q^H \tilde{M}\tilde{M}^H Q \end{array}\right]$	Create the matrix $\tilde{F}$, whose eigendecomposition will give us the SVD of $\tilde{M}'$. See eqns. (17) and (18).
3)	$\tilde{U}_f \tilde{\Sigma}_f \tilde{U}_f^H = \tilde{F}$	Calculate the eigendecomposition of $\tilde{F}$.
4)	$\tilde{U}' = [\, U' \mid Q \,]\tilde{U}_f$ $\tilde{\Sigma}'^2 = \tilde{\Sigma}_f$	Determine the singular values and left singular vectors of $\tilde{M}'$. See eqn. (9).

previous step of the algorithm, Σ' and U'. Remember that $\boldsymbol{x}_{t-c}$, the column that we are discarding, is the leftmost column of M, and $\boldsymbol{x}_t$, the column that we are adding, is the rightmost column of $\tilde{M}$.

Figure 1 shows the computation required to calculate the singular values and left singular vectors of a 64×64 matrix $\tilde{M}$. The two SVD methods (the purple and red lines) calculate the full set of singular values and left singular vectors of $\tilde{M}$. The two IFAST methods (the blue and green lines) calculate the full set of singular values and left singular vectors of $\tilde{M}'$, which are approximations to the $r+2$ largest singular values and left singular vectors of $\tilde{M}$. The two SVD methods will produce identical answers, and the two IFAST methods will produce identical answers.

The purple line represents calculating the full set of singular values and left

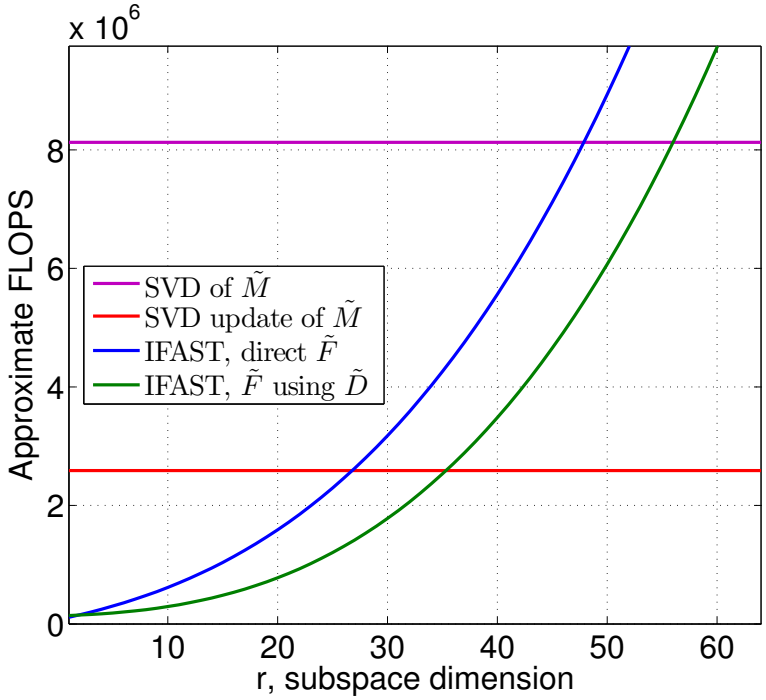


Figure 1. Comparison of the computation required (smaller is better) for a single IFAST iteration vs. computing the full SVD of $\tilde{M}$. For this figure, $n = c = 64$ with complex $\tilde{M}$. The two SVD methods will produce identical answers, and the two IFAST methods will produce identical answers.

singular vectors of $\tilde{M}$ without using any previous information. The red line represents calculating the full set of singular values and left singular vectors of $\tilde{M}$ by updating the singular values and left singular vectors of M using the method from manuscript 2. Both lines are flat because they are not a function of r. They both require $O(n^3)$ computation for square $\tilde{M}$, and their computation differs only by a constant scale factor.

The blue line represents calculating the singular values and left singular vectors of $\tilde{M}'$ using steps 1, 3, and 4 of the IFAST algorithm, but in step 2, $\tilde{F}$ is created directly using (8), as in $\tilde{F} = [\, U' \,|\, Q \,]^H \tilde{M} \tilde{M}^H [\, U' \,|\, Q \,]$. The green line represents calculating the singular values and left singular vectors of $\tilde{M}'$ using the IFAST algorithm as shown in Table 1. They both have an $O((r+2)^3)$ SVD of $\tilde{F}$ from step 3, which dominates for large r. The difference in computation comes from the

$r \times r$ upper left block of $\tilde{F}$, which is an $O(ncr) + O(cr^2)$ calculation for the direct method, and an $O(nr)$ calculation when using $\tilde{D}$.

1.2.5 Accuracy Example

The example in Figure 2 illustrates the accuracy of the IFAST algorithm. It compares the IFAST algorithm (the green line) with the FAST algorithm of Real, Tufts, and Cooley [1] (the blue line). The red line uses the SVD to produce U', (the singular vectors of M), for each iteration.

In this example, the data are generated from a single sensor, and consist of the sum of two complex chirps, one rectangularly windowed complex sinusoid, and complex white noise. The signal is generated from the equation

$$s(t) = s_1(t) + s_2(t) + s_3(t) + n(t), \qquad t = 0, 1, \ldots, 449, \tag{25}$$

where

$$s_1(t) = 1.19e^{j(2\pi/3 + t\pi/2048)t}, \tag{26}$$

$$s_2(t) = 1.1e^{j(5.5\pi/5 - t\pi/2048)t}, \tag{27}$$

$$s_3(t) = 1.23e^{j(\pi/3)t}u(t-100)u(349-t), \tag{28}$$

and $n(t)$ is zero mean, uncorrelated complex Gaussian white noise, with a variance of 1, (ie. $n(t) \sim CN(0,1)$), and $u(t)$ is the unit step function. The signal, $s(t)$, is then made into 387 overlapping 32×32 Hankel matrices, which are used by the various algorithms. This signal is similar to that used in experiment 4 from [1], but with a much lower SNR. The SNR for $s_1(t)+n(t)$, $s_2(t)+n(t)$, and $s_3(t)+n(t)$ are 1.5, 0.8, and 1.8 respectively. The spectrogram of $s(t)$ is shown in Figure 2e. Looking at the vertical normalized frequency axis, the signal $s_1(t)$ is the increasing line starting at about 0.33, the signal $s_2(t)$ is the decreasing line starting at about 0.55, and the signal $s_3(t)$ is the line at 0.17.

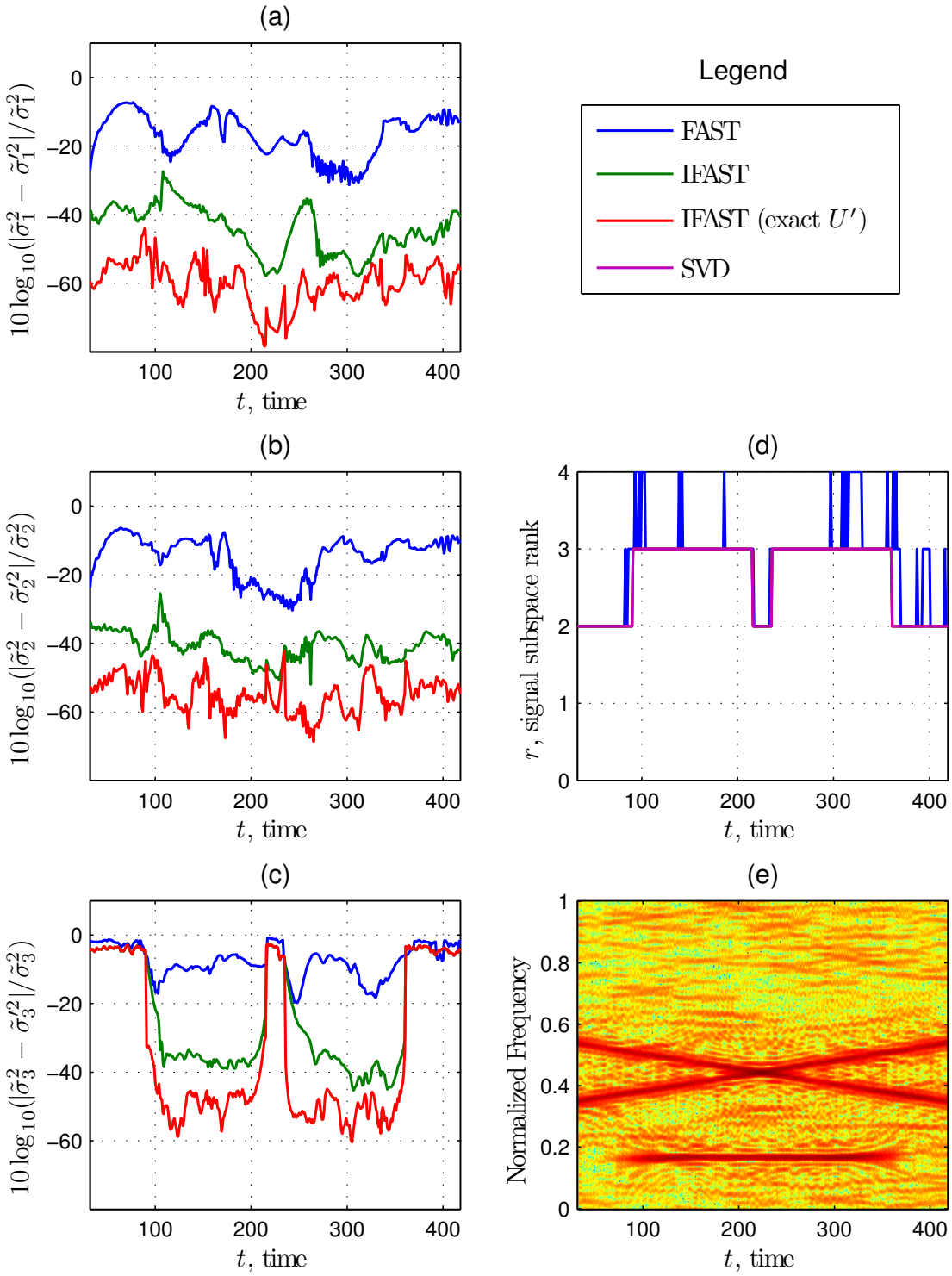


Figure 2. Normalized error in dB for the three largest singular values of $\tilde{M}$, which are $\tilde{\sigma}_1^2$, $\tilde{\sigma}_2^2$, and $\tilde{\sigma}_3^2$, along with the estimated signal subspace rank, and spectrogram of the signal, which is two complex chirps and a finite duration complex exponential.

The method from manuscript 4 and [6] was used to determine the number of signals present. The confidence level, α, was chosen to be 0.999. Figure 2d shows the rank estimates for all four methods. The two IFAST methods and the SVD all produced identical rank estimates, therefore the lines for the IFAST methods are under the line for the SVD. The rank estimates using the FAST algorithm deviated from the other methods a few times. The reason that the rank is estimated to be two instead of three when the chirps cross in frequency, is because the chirps become almost linearly dependent at that point.

Figure 2, subplots a, b, and c show the normalized error in dB for the first three singular value estimates ($\tilde{\sigma}'_1$, $\tilde{\sigma}'_2$, $\tilde{\sigma}'_3$) vs. the true singular values ($\tilde{\sigma}_1$, $\tilde{\sigma}_2$, $\tilde{\sigma}_3$). One immediate consequence of this, is that if we multiply the normalized error by 10 and negate it, this gives us the number of correct digits in our estimate. For instance, at time $t = 200$, the true value of $\tilde{\sigma}_1^2$ is 1300.916314593100, FAST estimated it to be 1210.752108327795, which has the first digit correct, IFAST estimated it to be 1300.793378303474, which has the first four digits correct, and IFAST using the exact U' estimated it to be 1300.915147265195, which has the first six digits correct. In Figure 2c, all methods do a pretty poor job of estimating the third largest singular value when we only have two columns in U'. This estimate quickly improves when r is at least three, which also means that we have at least three columns in U'.

1.2.6 Starting Without an Initial SVD, the IFAST-S Algorithm

The IFAST algorithm is intended to be used iteratively. It uses U' and Σ'^2 from the previous iteration to determine $\tilde{U}'$ and $\tilde{\Sigma}'^2$ for the current iteration. When the algorithm starts, we will not have U' and Σ'^2; therefore, we can either perform a full SVD on our first matrix M, or use a slightly modified version of the IFAST algorithm to determine the initial U' and Σ'^2. We will call this modified version of

Table 2. Steps of the IFAST-S (start-up) Algorithm

Step	
1)	$\boldsymbol{z}_1 = \left(I - U'U'^H\right)\boldsymbol{x}_t$ $Q = \boldsymbol{z}_1 / \left\|\boldsymbol{z}_1\right\|$
2)	$\tilde{D} = \Sigma'^2 + U'^H \boldsymbol{x}_t \boldsymbol{x}_t^H U'$ $\tilde{F} = \left[\begin{array}{c\|c} \tilde{D} & U'^H \tilde{M}\tilde{M}^H Q \\ \hline Q^H \tilde{M}\tilde{M}^H U' & Q^H \tilde{M}\tilde{M}^H Q \end{array}\right]$
3)	$\tilde{U}_f \tilde{\Sigma}_f \tilde{U}_f^H = \tilde{F}$
4)	$\tilde{U}' = [\, U' \mid Q \,]\tilde{U}_f$ $\tilde{\Sigma}'^2 = \tilde{\Sigma}_f$

the algorithm the *IFAST startup* algorithm, or the IFAST-S algorithm for short.

To use IFAST-S to determine the initial U' and Σ'^2, we start with a single column in M, which will be $\boldsymbol{x}_0$. We can write its singular value and left singular vector as $\Sigma' = \|\boldsymbol{x}_0\|$ and $U' = \boldsymbol{x}_0/\|\boldsymbol{x}_0\|$. For each iteration we add a new column to M, but do not remove any columns until M contains the desired number of columns. While M is growing, Q from Step 1 of the algorithm will have only one column, and $\tilde{F}$ will be an $r+1 \times r+1$ matrix. The steps of the IFAST-S algorithm are given in Table 2.

Whether to use an initial SVD or the IFAST-S algorithm depends on how the IFAST algorithm is being used. The IFAST-S algorithm requires more overall computation than the SVD, but has no latency. Figure 3 illustrates how using the two methods differ for a 64×64 matrix M. We can start IFAST-S computations right when we get the first column at time t_0, and when we receive the 64th column at time t_{63}, we can determine our estimates of of Σ'^2 and U' in the time it takes to do a typical IFAST iteration. The SVD cannot start computations until after the 64th column is received, therefore it will take more than the typical time for a

single IFAST iteration to complete.

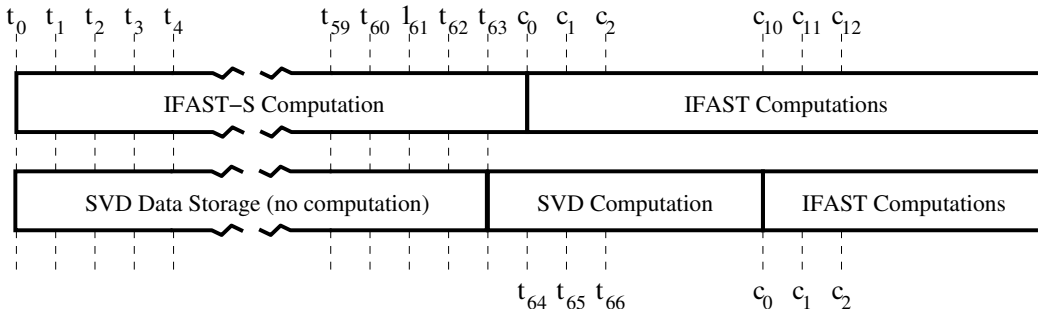


Figure 3. An illustration to show how determining the initial Σ'^2 and U' using the IFAST-S algorithm requires more overall computation, but has no latency, whereas using the SVD is more computationally efficient, but has large latency due to the fact that it cannot start computation until the last column is recieved.

Generally, when all the columns are available at once, and the algorithm is not being implemented on special purpose hardware, it makes sense to use an initial SVD to determine the initial Σ'^2 and U', because neither latency nor a large SVD implementation will be an issue. In a real-time system, where the columns will often be received one at a time, and possibly special purpose hardware will be used, the IFAST-S algorithm makes more sense because its implementation is almost identical to that of IFAST, and it has no latency.

1.2.7 Adding More than One Column per Iteration, the IFAST-M Algorithm

When more than one new column is added and removed each iteration, it is possible to make a simple modification to the algorithm to account for this. We will call this modified version of the algorithm the *IFAST Multicolumn* algorithm, or the IFAST-M algorithm for short. The orthonormal matrix Q is generated using a Gram-Schmidt method, and contains the column space spanning the columns we are adding and the columns that we are removing, that is orthogonal to U'. In other words, Q will satisfy $[\,U' \,|\, Q\,]^H [\,U' \,|\, Q\,] = I$, no matter how many columns it contains, up to $n-r$ columns. If k new columns are being added per iteration, and

Table 3. Steps of the IFAST-M (multi-column) Algorithm

Step	
1)	$Q = [\ ]$ for $i = 0 \cdots k-1$, $\boldsymbol{z} = \left(I - [\,U' \mid Q\,]\,[\,U' \mid Q\,]^H\right)\boldsymbol{x}_{t-i}$ $Q = [\,Q \mid \boldsymbol{z}/\,\|\boldsymbol{z}\|\,]$ $\boldsymbol{z} = \left(I - [\,U' \mid Q\,]\,[\,U' \mid Q\,]^H\right)\boldsymbol{x}_{t-c-i}$ $Q = [\,Q \mid \boldsymbol{z}/\,\|\boldsymbol{z}\|\,]$ end
2)	$\tilde{D} = \Sigma'^2 - U'^H \boldsymbol{x}_{t-c}\boldsymbol{x}_{t-c}^H U' + U'^H \boldsymbol{x}_t \boldsymbol{x}_t^H U'$ $\tilde{F} = \left[\begin{array}{c\|c} \tilde{D} & U'^H \tilde{M}\tilde{M}^H Q \\ \hline Q^H \tilde{M}\tilde{M}^H U' & Q^H \tilde{M}\tilde{M}^H Q \end{array}\right]$
3)	$\tilde{U}_f \tilde{\Sigma}_f \tilde{U}_f^H = \tilde{F}$
4)	$\tilde{U}' = [\,U' \mid Q\,]\tilde{U}_f$ $\tilde{\Sigma}'^2 = \tilde{\Sigma}_f$

the number of columns in M is kept the same, the matrix Q will have dimensions $n \times 2k$, and the matrix $\tilde{F}$ will have dimensions $(r+2k) \times (r+2k)$. The only modification to the IFAST algorithm is in Step 1 of Table 1, where Q will now contain one column for each column being added, and one column for each column being discarded. The steps of the IFAST-M algorithm are given in Table 3. It will become more efficient to perform a full SVD on $\tilde{M}$ when $(r+2k)$ approaches n.

1.2.8 Taking Advantage of Hankel Structure, the IFAST-F Algorithm

When the matrix M has Hankel structure, and $c \ll n$, the calculations required to construct $\tilde{F}$ can be further reduced using the Fast Fourier Transform [9, 10]. This has been presented for the original version of FAST in manuscript 3 and [11], and can easily be applied to the IFAST algorithm. We will call this modi-

fied version of the algorithm the *IFAST FFT* algorithm, or the IFAST-F algorithm for short.

The steps of the algorithm are presented in Table 4. The only step that differs from the IFAST algorithm is Step 2, which is the construction of $\tilde{F}$. The matrix Υ is an $\hat{N} \times (r+2)$ matrix, where $\hat{N} = n + c - 1$. The matrix $\boldsymbol{d}_t$ is a length $\hat{N}$ column vector consisting of the unique elements that were used to construct the Hankel matrix $\tilde{M}$. The "$\odot$" operator is the Hadamard product, meaning element by element multiplication of the matrices. After taking the inverse FFT of E, we only keep the first c columns, leaving the $(r+2) \times c$ matrix $\tilde{E}$. The $O(c(r+2)^2)$ matrix product $\tilde{E}\tilde{E}^H$, will be the dominant computation, therefore c will be what determines whether the IFAST or the IFAST-S algorithm is more efficient.

1.2.9 General Comments About the Algorithm

The IFAST-M algorithm and the IFAST-F algorithm can easily be merged to create an algorithm which can be used when adding and removing multiple columns of a Hankel matrix per iteration. This combined algorithm will only be more efficient than the IFAST-M algorithm when $c \ll n$ though. The IFAST-S algorithm can easily be modified to be a startup algorithm for any of the other algorithms.

In the introduction, we mentioned that IFAST is more computationally efficient that the FAST algorithm. The computation required for the FAST algorithm is very close to the blue curve in Figure 1. The computational efficiency in IFAST comes from creating the matrix $\tilde{F}$, by first creating the matrix $\tilde{D}$ as the sum of the old singular values and two rank-one matrices. This technique can be applied to the FAST algorithm, but it is more complex because $\tilde{D}$ would be the sum of the old singular values and five rank one matrices due to the forced zeros in the matrix $(r+1)$th column, even though there is only one column in Q.

Table 4. Steps of the IFAST-F Algorithm

Step		Description
1)	$\boldsymbol{z}_1 = \left(I - U'U'^H\right)\boldsymbol{x}_t$ $\boldsymbol{q}_1 = \boldsymbol{z}_1 / \left\|\boldsymbol{z}_1\right\|$ $\boldsymbol{z}_2 = \left(I - \left[\, U' \mid \boldsymbol{q}_1 \,\right]\left[\, U' \mid \boldsymbol{q}_1 \,\right]^H\right)\boldsymbol{x}_{t-c}$ $\boldsymbol{q}_2 = \boldsymbol{z}_2 / \left\|\boldsymbol{z}_2\right\|$ $Q = \left[\, \boldsymbol{q}_1 \mid \boldsymbol{q}_2 \,\right]$	Identical to Step 1 of IFAST.
2)	$\Upsilon \xleftrightarrow{\text{FFT}} \left[\begin{array}{c\|c} U' & Q \\ \hline 0 & \end{array}\right]$ $\boldsymbol{\delta} \xleftrightarrow{\text{FFT}} \boldsymbol{d}_t$ $E = \Upsilon^H \odot \left[\, \boldsymbol{\delta} \;\cdots\; \boldsymbol{\delta} \,\right]^T$ $\left[\; \tilde{E} \;\middle|\; \tilde{\boldsymbol{e}}_{c+1} \;\; \cdots \;\; \tilde{\boldsymbol{e}}_N \;\right] \xleftrightarrow{\text{IFFT}} E$ $\tilde{F} = \tilde{E}\tilde{E}^H$	Take the FFT of each column of $\left[\begin{smallmatrix} U' & Q \\ 0 & 0 \end{smallmatrix}\right]$ create to Υ, and take the FFT of $\boldsymbol{d}_t$ to create $\boldsymbol{\delta}$. Create E by taking the Hadamard product of Υ^H and $\boldsymbol{\delta}^T$ replicated $r+2$ times. Create $\tilde{E}$ by taking the first c columns of the inverse FFT of E. Create $\tilde{F}$ by multiplying $\tilde{E}$ by $\tilde{E}^H$.
3)	$\tilde{U}_f \tilde{\Sigma}_f \tilde{U}_f^H = \tilde{F}$	Identical to Step 3 of IFAST.
4)	$\tilde{U}' = \left[\, U' \mid Q \,\right]\tilde{U}_f$ $\tilde{\Sigma}'^2 = \tilde{\Sigma}_f$	Identical to Step 4 of IFAST.

The eigendecomposition of $\tilde{F}$ in Step 3 of the IFAST algorithm is generally the most computationally dominant step. When there are hardware limitations that only allow an SVD of a certain size, say r_{max}, the algorithm will still work. What will happen, is that while the signal subspace rank is greater than r_{max}, the algorithm will produce r_{max} singular value and singular vector estimates. Because we are not able to track part of the signal subspace, that part will end up in the noise subspace, and effectively decrease the SNR. When the signal subspace returns below r_{max}, the algorithm will again track the full signal subspace.

The matrix Q is not unique, and in Step 1 of IFAST, we could have created $\boldsymbol{q}_1$ using $\boldsymbol{x}_{t-c}$ and $\boldsymbol{q}_2$ using $\boldsymbol{x}_t$ instead. In fact, we can see from (6) that any rotation

of Q by a 2×2 unitary matrix will have no effect on the algorithm, because the $[\, U' \,|\, Q \,][\, U' \,|\, Q \,]^H$ term will multiply this 2×2 unitary matrix by its Hermitian, giving a 2×2 identity matrix.

It should be noted that the order of evaluation is important in efficiently performing the calculations, for example $\boldsymbol{z}_1$ in step one of the IFAST algorithm might be evaluated as $\boldsymbol{z}_1 = \boldsymbol{x}_t - U'(U'^H \boldsymbol{x}_t)$.

It should be noted that the IFAST algorithm works for both complex and real data, whereas the rank tracking method from manuscript 4 and [6] is designed for complex data only. The IFAST algorithm is designed to update the singular values and singular vectors of M; thus, it is independent of the rank tracking algorithm. Therefore, any rank tracking method can be used.

1.3 Rank-Two Secular Functions

In this section, we will rewrite the SVD of $\tilde{M}$ and the SVD of $\tilde{M}'$, as equivalent problems which are the eigendecomposition of a diagonal matrix plus two rank-one matrices. This allows us to analyze them using the rank-two secular function, whose roots are the eigenvalues of the matrix. The rank-two secular function is presented in manuscript 2.

When a matrix can be written in the form $D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H$, where D is a diagonal matrix, and $\boldsymbol{a}$ and $\boldsymbol{b}$ are vectors, the eigenvalues of this matrix are the roots of the rank-two secular function. The secular function [12] is closely related to the characteristic polynomial [13], but has some qualities which make it more useful for analysis.

1.3.1 The Full Dimension, $\tilde{M} \Rightarrow \tilde{G}$

We will start off with the two $n \times c$ overlapping matrices from (2) and (3),

$$\begin{aligned} M &= \left[\begin{array}{cccc} \boldsymbol{x}_{t-c} & \boldsymbol{x}_{t-c+1} & \boldsymbol{x}_{t-c+2} & \cdots & \boldsymbol{x}_{t-1} \end{array} \right], \\ \tilde{M} &= \left[\begin{array}{cccc} \boldsymbol{x}_{t-c+1} & \boldsymbol{x}_{t-c+2} & \cdots & \boldsymbol{x}_{t-1} & \boldsymbol{x}_{t} \end{array} \right], \end{aligned}$$

along with the SVD of $M = U\Sigma V^H$, where $U \in \mathbb{C}^{n \times n}$ and $\Sigma \in \mathbb{R}^{n \times c}$. If we define $\tilde{G}$ as $U^H \tilde{M} \tilde{M}^H U$, then from (22), we can write

$$\tilde{G} = U^H M M^H U - U^H \boldsymbol{x}_{t-c} \boldsymbol{x}_{t-c}^H U + U^H \boldsymbol{x}_t \boldsymbol{x}_t^H U. \tag{29}$$

If we define the two, length n, column vectors

$$\boldsymbol{a} = U^H \boldsymbol{x}_{t-c} \qquad \text{and} \qquad \boldsymbol{b} = U^H \boldsymbol{x}_t, \tag{30}$$

and zero pad Σ so that we can write $\Sigma^2 \in \mathbb{R}^{n \times n}$, then we can write (29) as

$$\tilde{G} = \Sigma^2 - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H, \tag{31}$$

which is a diagonal matrix plus two rank one matrices. If we write the eigendecomposition of $\tilde{G}$ as $U_g \Lambda_g U_g^H$, then from Theorem 1, we can write the singular values and left singular vectors of $\tilde{M}$ as

$$\tilde{\Sigma} = \sqrt{\Lambda_g} \qquad \text{and} \qquad \tilde{U} = U U_g. \tag{32}$$

What this means, is that we can analyze the singular values and left singular vectors of $\tilde{M}$ by analyzing the eigendecomposition of $\tilde{G}$.

The eigenvalues of $\tilde{G}$ are the roots of the rank-two secular equation

$$w_g(\lambda) = \left(1 - \sum_{j=1}^{n} \frac{|a_j|^2}{\sigma_j^2 - \lambda}\right)\left(1 + \sum_{j=1}^{n} \frac{|b_j|^2}{\sigma_j^2 - \lambda}\right) + \left|\sum_{j=1}^{n} \frac{a_j^* b_j}{\sigma_j^2 - \lambda}\right|^2, \tag{33}$$

$$w_g(\lambda) = w_{g_a}(\lambda) w_{g_b}(\lambda) + |w_{g_x}(\lambda)|^2, \tag{34}$$

where $w_{g_a}(\lambda)$ is the rank-one secular equation for $\Sigma^2 - \boldsymbol{a}\boldsymbol{a}^H$, and $w_{g_b}(\lambda)$ is the rank-one secular equation for $\Sigma^2 + \boldsymbol{b}\boldsymbol{b}^H$, [12, 14], and $w_{g_x}(\lambda)$ contains the cross terms.

1.3.2 The Principal Subspace, $U'U'^H\tilde{M} \Rightarrow \tilde{D}$

If we separate M into a strong r dimensional principal subspace, and an $n-r$ dimensional weak orthogonal subspace, we can write the SVD of M as

$$M = \begin{bmatrix} U' & U^\perp \end{bmatrix} \begin{bmatrix} \Sigma' & 0 \\ 0 & \Sigma^\perp \end{bmatrix} \begin{bmatrix} V' & V^\perp \end{bmatrix}^H, \tag{35}$$

where $\Sigma' \in \mathbb{R}^{r\times r}$ contains the r largest singular values of M, and $U' \in \mathbb{C}^{n\times r}$ contains the first r columns of U. We now want to analyze the SVD of

$$\tilde{M}^\ddagger = U'U'^H\tilde{M} = \tilde{U}^\ddagger\tilde{\Sigma}^\ddagger\tilde{V}^{\ddagger H}. \tag{36}$$

If we define $\tilde{D}$ to be $U'^H\tilde{M}\tilde{M}^HU'$, then from (17), we can write $\tilde{D}$ as

$$\tilde{D} = \Sigma'^2 - \boldsymbol{a}'\boldsymbol{a}'^H + \boldsymbol{b}'\boldsymbol{b}'^H, \tag{37}$$

where $\boldsymbol{a}'$ and $\boldsymbol{b}'$ are length r vectors that are just the first r elements of $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively. If we write the eigendecomposition of $\tilde{D}$ as $U_d\Lambda_dU_d^H$, then from Theorem 1 we can write the singular values and left singular vectors of $\tilde{M}^\ddagger$ as

$$\tilde{\Sigma}^\ddagger = \sqrt{\Lambda_d} \qquad \text{and} \qquad \tilde{U}^\ddagger = U'U_d. \tag{38}$$

What this means, is that we can analyze the singular values and left singular vectors of $\tilde{M}^\ddagger$ by analyzing the the eigendecomposition of $\tilde{D}$.

The eigenvalues of $\tilde{D}$ are the roots of the rank-two secular equation

$$w_d(\lambda) = \left(1 - \sum_{j=1}^{r} \frac{|a_j|^2}{\sigma_j^2 - \lambda}\right)\left(1 + \sum_{j=1}^{r} \frac{|b_j|^2}{\sigma_j^2 - \lambda}\right) + \left|\sum_{j=1}^{r} \frac{a_j^* b_j}{\sigma_j^2 - \lambda}\right|^2. \tag{39}$$

Note that (39) differs from (33) only by the upper limit of the summation.

1.3.3 The IFAST Subspace, $\tilde{M}' \Rightarrow \tilde{F}$

We now want to analyze the SVD of the IFAST approximation from (6),

$$\tilde{M}' = [\, U' \,|\, Q \,][\, U' \,|\, Q \,]^H\tilde{M} = \tilde{U}'\tilde{\Sigma}'\tilde{V}'^H. \tag{40}$$

In this section, we will put an additional constraint on Q, such that $Q^H M M^H Q$ is diagonal. It was shown in section 1.2.9 that multiplying Q by a unitary 2×2 matrix has no effect on the algorithm. If we create an initial Q using (5), which we will call Q_i, then take the SVD of the 2×2 matrix $Q_i^H M M^H Q_i = \hat{U}\hat{\Sigma}\hat{U}^H$, we can define $Q = Q_i \hat{U}$. This will give us

$$\hat{\Sigma} = Q^H M M^H Q, \tag{41}$$

which is a diagonal matrix. We can now write the lower right 2×2 block of $\tilde{F}$ from (18) as a diagonal matrix plus two rank one matrices of the form

$$Q^H \tilde{M} \tilde{M}^H Q = \hat{\Sigma} - \hat{\boldsymbol{a}}\hat{\boldsymbol{a}}^H + \hat{\boldsymbol{b}}\hat{\boldsymbol{b}}^H, \tag{42}$$

where

$$\hat{\boldsymbol{a}} = Q^H \boldsymbol{x}_{t-c} \qquad \text{and} \qquad \hat{\boldsymbol{b}} = Q^H \boldsymbol{x}_t. \tag{43}$$

Combining eqns. (18), (37), (42), and (22), we can write

$$\tilde{F} = \left[\begin{array}{c|c} \Sigma'^2 & U'^H M M^H Q \\ \hline Q^H M M^H U' & \hat{\Sigma} \end{array}\right] - \left[\begin{array}{c} \boldsymbol{a}' \\ \hline \hat{\boldsymbol{a}} \end{array}\right] \left[\begin{array}{c} \boldsymbol{a}' \\ \hline \hat{\boldsymbol{a}} \end{array}\right]^H + \left[\begin{array}{c} \boldsymbol{b}' \\ \hline \hat{\boldsymbol{b}} \end{array}\right] \left[\begin{array}{c} \boldsymbol{b}' \\ \hline \hat{\boldsymbol{b}} \end{array}\right]^H. \tag{44}$$

Because we are assuming U' consists of the r true left singular vectors of M (as opposed to approximate left singular vectors), then from (35), we can write $M M^H = U' \Sigma'^2 U'^H + U^\perp \Sigma^{\perp 2} U^{\perp H}$, which when substituted into $Q^H M M^H U'$, gives us

$$Q^H M M^H U' = Q^H \left(U' \Sigma'^2 U'^H + U^\perp \Sigma^{\perp 2} U^{\perp H} \right) U'. \tag{45}$$

The first term is zero because $Q^H U' = 0$, and the second term is zero because $U^{\perp H} U' = 0$. This allows us to write $\tilde{F}$ as a diagonal matrix plus two rank one matrices of the form,

$$\tilde{F} = \left[\begin{array}{c|c} \Sigma'^2 & 0 \\ \hline 0 & \hat{\Sigma} \end{array}\right] - \left[\begin{array}{c} \boldsymbol{a}' \\ \hline \hat{\boldsymbol{a}} \end{array}\right] \left[\begin{array}{c} \boldsymbol{a}' \\ \hline \hat{\boldsymbol{a}} \end{array}\right]^H + \left[\begin{array}{c} \boldsymbol{b}' \\ \hline \hat{\boldsymbol{b}} \end{array}\right] \left[\begin{array}{c} \boldsymbol{b}' \\ \hline \hat{\boldsymbol{b}} \end{array}\right]^H. \tag{46}$$

If we write the eigendecomposition of $\tilde{F}$ as $U_f \Lambda_f U_f^H$, then from Theorem 1, we can write the singular values and left singular vectors of $\tilde{M}'$ as

$$\tilde{\Sigma}' = \sqrt{\Lambda_f} \qquad \text{and} \qquad \tilde{U}' = [\, U' \,|\, Q \,] U_f. \tag{47}$$

What this means, is that we can analyze the singular values and left singular vectors of $\tilde{M}'$ by analyzing the eigendecomposition of $\tilde{F}$.

The eigenvalues of $\tilde{F}$ are the roots of the rank-two secular equation

$$\begin{aligned} w_f(\lambda) &= \left(1 - \sum_{j=1}^{r} \frac{|a_j|^2}{\sigma_j^2 - \lambda} - \sum_{j=1}^{2} \frac{|\hat{a}_j|^2}{\hat{\sigma}_j - \lambda}\right)\left(1 + \sum_{j=1}^{r} \frac{|b_j|^2}{\sigma_j^2 - \lambda} + \sum_{j=1}^{2} \frac{|\hat{b}_j|^2}{\hat{\sigma}_j - \lambda}\right) \\ &\qquad + \left|\sum_{j=1}^{r} \frac{a_j^* b_j}{\sigma_j^2 - \lambda} + \sum_{j=1}^{2} \frac{\hat{a}_j^* \hat{b}_j}{\hat{\sigma}_j - \lambda}\right|^2, \qquad (48) \\ w_f(\lambda) &= w_{f_a}(\lambda) w_{f_b}(\lambda) + |w_{f_x}(\lambda)|^2, \qquad (49) \end{aligned}$$

where $\hat{\sigma}_j$ comes from the diagonal of $\hat{\Sigma}$ in (41), and $\hat{a}_i$ and $\hat{b}_i$ are the i elements of $\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{b}}$ respectively.

1.3.4 Secular Function Comparison

Figure 4 shows an example of the rank-two secular functions for $\tilde{G}$, $\tilde{D}$, and $\tilde{F}$, for (31), (37), and (46) respectively. The roots of the secular function are the eigenvalues of the respective matrix, $\tilde{G}$, $\tilde{D}$, or $\tilde{F}$, which are the squares of the singular values of the matrices $\tilde{M}$, $\tilde{M}^{\ddagger}$, or $\tilde{M}'$. The poles of the secular function are the eigenvalues of the original unperturbed matrix, which is the diagonal matrix in $\tilde{G}$, $\tilde{D}$, or $\tilde{F}$. Here it is assumed that $n = c = 20$, $r = 7$, and the SNR=4. The function $w_g(\lambda)$ has 20 poles at $\sigma_1^2 \cdots \sigma_{20}^2$, and 20 roots at $\tilde{\sigma}_1^2 \cdots \tilde{\sigma}_{20}^2$. The function $w_d(\lambda)$ has 7 poles at $\sigma_1^2 \cdots \sigma_7^2$, and 7 roots at $\tilde{\sigma}_1^{\ddagger 2} \cdots \tilde{\sigma}_7^{\ddagger 2}$. The function $w_f(\lambda)$ has 9 poles at $\sigma_1^2 \cdots \sigma_7^2$, $\hat{\sigma}_1$ and $\hat{\sigma}_2$, and 9 roots at $\tilde{\sigma}_1'^2 \cdots \tilde{\sigma}_9'^2$. Note that σ_1^2 through σ_7^2 are poles for all three secular functions, and since $w_f(\lambda)$ was plotted last in Figure 4, we see red vertical lines at these points.

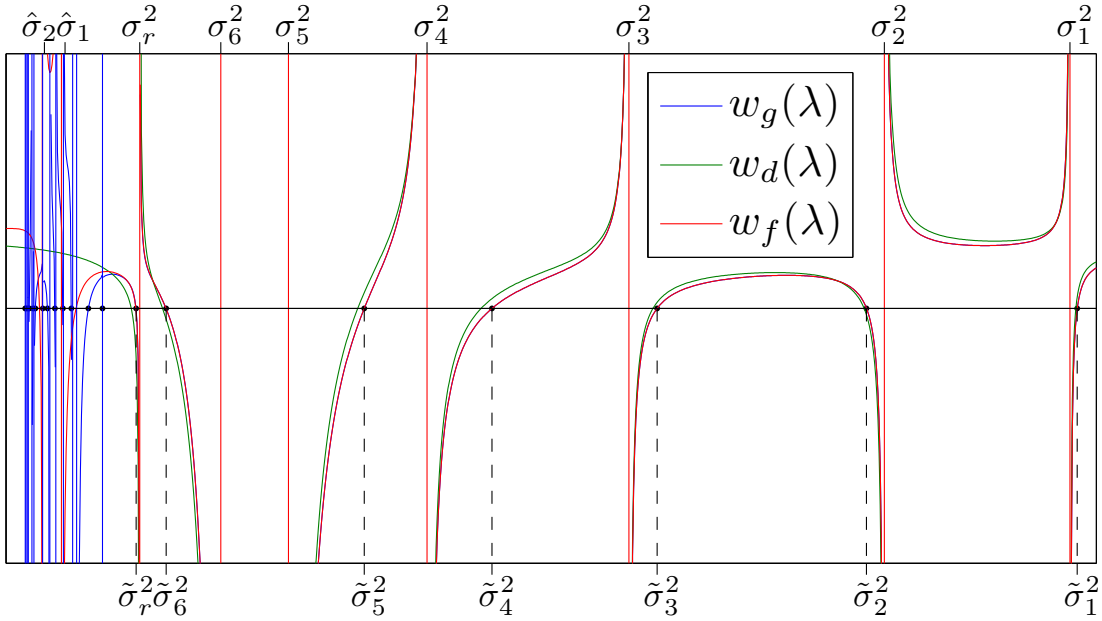


Figure 4. An example of the rank-two secular functions $w_g(\lambda)$, $w_d(\lambda)$, and $w_f(\lambda)$, from (33), (39), and (48), vs. λ for $n = c = 20$, and principal subspace $r = 7$. SNR is 4. The poles, σ_i^2 indicated at the top of the plot, correspond to the old singular values, and the roots, $\tilde{\sigma}_i^2$ indicated at the bottom of the plot, correspond to the new singular values.

The reason that $w_g(\lambda)$ is not visible for $\lambda > \tilde{\sigma}_r^2$, is because it is concealed by $w_f(\lambda)$ in that region. For $\lambda < \tilde{\sigma}_r^2$, there are 2 roots and poles for $w_f(\lambda)$, and $n - r = 13$ roots and poles for $w_g(\lambda)$, which correspond to the rest of the singular values of M and $\tilde{M}$.

The only difference between $w_g(\lambda)$ and $w_d(\lambda)$ is the upper limit of their summations. This means that terms for the contribution for the r largest singular values are represented exactly, while the terms for the smaller singular values are completely unrepresented. The function $w_f(\lambda)$ tries to represent the $n-r$ smallest poles of $w_g(\lambda)$ by the two poles $\hat{\sigma}_1$ and $\hat{\sigma}_2$.

1.4 The Rank-One Case

Our goal is to analyze the difference between the eigendecomposition of $\tilde{G}$, which will give us the true SVD of $\tilde{M}$, and the eigendecomposition of $\tilde{F}$, which

will give us the IFAST approximation to the SVD of $\tilde{M}$, which is the true SVD of $\tilde{M}'$. The rank two modification of the diagonal matrix in $\tilde{G}$ and $\tilde{F}$ can be analyzed as two sequential rank one modifications. Because the math is easier to follow using the rank-one secular equation, we will first analyze the rank one modification which comes from the column we are adding to M, that is, $\boldsymbol{x}_t$. We can write this modification as

$$\tilde{G}_b = \Sigma^2 + \boldsymbol{b}\boldsymbol{b}^H, \tag{50}$$

$$\tilde{F}_b = \left[\begin{array}{c|c} \Sigma'^2 & 0 \\ \hline 0 & \hat{\Sigma} \end{array}\right] + \left[\begin{array}{c} \boldsymbol{b}' \\ \hline \hat{\boldsymbol{b}} \end{array}\right] \left[\begin{array}{c} \boldsymbol{b}' \\ \hline \hat{\boldsymbol{b}} \end{array}\right]^H. \tag{51}$$

Their rank-one secular functions from (34) and (49), whose roots are the eigenvalues of $\tilde{G}_b$ and $\tilde{F}_b$, are

$$w_{g_b}(\lambda) = 1 + \sum_{j=1}^{n} \frac{|b_j|^2}{\sigma_j^2 - \lambda}, \tag{52}$$

$$w_{f_b}(\lambda) = 1 + \sum_{j=1}^{r} \frac{|b_j|^2}{\sigma_j^2 - \lambda} + \sum_{j=1}^{2} \frac{|\hat{b}_j|^2}{\hat{\sigma}_j - \lambda}. \tag{53}$$

Since roots of $w_{g_b}(\lambda)$ will give us the true singular values of the first rank-one update of $\tilde{M}$, we can think of $w_{f_b}(\lambda)$ as equal to $w_{g_b}(\lambda)$ plus an error term. Therefore, we can write

$$w_{f_b}(\lambda) = w_{g_b}(\lambda) + e_{f_b}(\lambda), \tag{54}$$

where $e_{f_b}(\lambda)$ is the error term. From (52) and (53), we can write

$$e_{f_b}(\lambda) = w_{f_b}(\lambda) - w_{g_b}(\lambda) = \sum_{j=1}^{2} \frac{|\hat{b}_j|^2}{\hat{\sigma}_j - \lambda} - \sum_{j=r+1}^{n} \frac{|b_j|^2}{\sigma_j^2 - \lambda}. \tag{55}$$

1.4.1 Infinite Geometric Series Expansions

We will now perform infinite geometric series expansions on the two summations in (55) to show that the first two terms in both expansions are identical. For λ greater than $\max(\sigma_{r+1}, \hat{\sigma}_1)$, we can use an infinite geometric series expansion to

write

$$\sum_{j=r+1}^{n} \frac{|b_j|^2}{\sigma_j^2 - \lambda} = -\sum_{j=r+1}^{n} \frac{|b_j|^2}{\lambda} \left(\frac{1}{1 - \sigma_j^2/\lambda} \right) = -\sum_{j=r+1}^{n} \frac{|b_j|^2}{\lambda} \sum_{i=0}^{\infty} \left(\frac{\sigma_j^2}{\lambda} \right)^i \tag{56}$$

and

$$\sum_{j=1}^{2} \frac{|\hat{b}_j|^2}{\hat{\sigma}_j - \lambda} = -\sum_{j=1}^{2} \frac{|\hat{b}_j|^2}{\lambda} \left(\frac{1}{1 - \hat{\sigma}_j/\lambda} \right) = -\sum_{j=1}^{2} \frac{|\hat{b}_j|^2}{\lambda} \sum_{i=0}^{\infty} \left(\frac{\hat{\sigma}_j}{\lambda} \right)^i. \tag{57}$$

For both of the series expansions, we will move the first two terms of the sum over i out front, to get

$$-\frac{1}{\lambda} \left(\sum_{j=r+1}^{n} |b_j|^2 \right) - \frac{1}{\lambda^2} \left(\sum_{j=r+1}^{n} |b_j|^2 \sigma_j^2 \right) - \left(\sum_{j=r+1}^{n} \frac{|b_j|^2}{\lambda} \sum_{i=2}^{\infty} \left(\frac{\sigma_j^2}{\lambda} \right)^i \right) \tag{58}$$

and

$$-\frac{1}{\lambda} \left(\sum_{j=1}^{2} |\hat{b}_j|^2 \right) - \frac{1}{\lambda^2} \left(\sum_{j=1}^{2} |\hat{b}_j|^2 \hat{\sigma}_j \right) - \left(\sum_{j=1}^{2} \frac{|\hat{b}_j|^2}{\lambda} \sum_{i=2}^{\infty} \left(\frac{\hat{\sigma}_j}{\lambda} \right)^i \right). \tag{59}$$

The first term from the series expansion in (58) can be written as

$$\sum_{j=r+1}^{n} |b_j|^2 = \sum_{j=r+1}^{n} \boldsymbol{x}_t^H \boldsymbol{u}_j \boldsymbol{u}_j^H \boldsymbol{x}_t = \boldsymbol{x}_t^H U^{\perp} U^{\perp H} \boldsymbol{x}_t. \tag{60}$$

Because of the way Q was constructed, $QQ^H \boldsymbol{x}_t = (I - U'U'^H)\boldsymbol{x}_t = U^{\perp} U^{\perp H} \boldsymbol{x}_t$. Therefore, (60) can be written as $\boldsymbol{x}_t^H QQ^H \boldsymbol{x}_t$, which is identical to the first term of (59), $\sum_{j=1}^{2} |\hat{b}_j|^2$.

The second term from the series expansion in (58) can be written as

$$\sum_{j=r+1}^{n} |b_j|^2 \sigma_j^2 = \sum_{j=r+1}^{n} \boldsymbol{x}_t^H \boldsymbol{u}_j \sigma_j^2 \boldsymbol{u}_j^H \boldsymbol{x}_t = \boldsymbol{x}_t^H U^{\perp} \Sigma^{\perp 2} U^{\perp H} \boldsymbol{x}_t. \tag{61}$$

We can insert the identity matrix, $U^{\perp H} U^{\perp}$, into (61) to get

$$\boldsymbol{x}_t^H U^{\perp} U^{\perp H} U^{\perp} \Sigma^{\perp 2} U^{\perp H} U^{\perp} U^{\perp H} \boldsymbol{x}_t. \tag{62}$$

Substituting $QQ^H \boldsymbol{x}_t = U^{\perp} U^{\perp H} \boldsymbol{x}_t$, we get

$$\boldsymbol{x}_t^H QQ^H U^{\perp} \Sigma^{\perp 2} U^{\perp H} QQ^H \boldsymbol{x}_t, \tag{63}$$

which is equal to

$$\boldsymbol{x}_t^H QQ^H(U'\Sigma'^2U'^H + U^\perp\Sigma^{\perp 2}U^{\perp H})QQ^H\boldsymbol{x}_t = \boldsymbol{x}_t^H QQ^H U\Sigma^2U^H QQ^H\boldsymbol{x}_t \tag{64}$$

because Q is orthogonal to the columns of U'. This can be written as

$$\boldsymbol{x}_t^H QQ^H MM^H QQ^H\boldsymbol{x}_t. \tag{65}$$

Finally, since $Q^H MM^H Q = \hat{\Sigma}$, we get

$$\boldsymbol{x}_t^H Q\hat{\Sigma}Q^H\boldsymbol{x}_t, \tag{66}$$

which is the same as the second term from (59), $\sum_{j=1}^{2}|\hat{b}_j|^2\hat{\sigma}_j$.

Combining (58) and (59) into (55), we get

$$e_{f_b}(\lambda) = \sum_{j=r+1}^{n}\frac{|b_j|^2}{\lambda}\sum_{i=2}^{\infty}\left(\frac{\sigma_j^2}{\lambda}\right)^i - \sum_{j=1}^{2}\frac{|\hat{b}_j|^2}{\lambda}\sum_{i=2}^{\infty}\left(\frac{\hat{\sigma}_j}{\lambda}\right)^i, \tag{67}$$

which can be written in terms of powers of singular values as

$$e_{f_b}(\lambda) = \sum_{i=2}^{\infty}\boldsymbol{x}_t^H Q\left(\frac{\hat{\Sigma}^i - Q^H U^\perp\Sigma^{\perp 2i}U^{\perp H}Q}{\lambda^{i+1}}\right)Q^H\boldsymbol{x}_t, \tag{68}$$

or in terms of powers of M as

$$e_{f_b}(\lambda) = \sum_{i=2}^{\infty}\boldsymbol{x}_t^H Q\left(\frac{(Q^H MM^H Q)^i - Q^H(MM^H)^iQ}{\lambda^{i+1}}\right)Q^H\boldsymbol{x}_t. \tag{69}$$

What we see is that the IFAST approximation uses the only Q matrix that exactly cancels the first two terms of the infinite geometric series expansion.

1.4.2 Separation Into Parts

Now that we have a term for the difference between $w_{g_b}(\lambda)$ and $w_{f_b}(\lambda)$, we will now define the common and unique parts of each function. We will write

$$w_{g_b}(\lambda) = c_b(\lambda) + u_{g_b}(\lambda), \tag{70}$$

$$w_{f_b}(\lambda) = c_b(\lambda) + u_{f_b}(\lambda), \tag{71}$$

where $c_b(\lambda)$ is the part of $w_{g_b}(\lambda)$ and $w_{f_b}(\lambda)$ that is common to both, $u_{g_b}(\lambda)$ is the part of $w_{g_b}(\lambda)$ that unique to it, and $u_{f_b}(\lambda)$ is the part of $w_{f_b}(\lambda)$ that is unique to it.

We know from (52) and (53) both $w_{g_b}(\lambda)$ and $w_{f_b}(\lambda)$ contain the term $1 + \sum_{j=1}^{r} \frac{|b_j|^2}{\sigma_j^2 - \lambda}$. Therefore, if we add the two terms from (58) and (59) that are equal, we come up with

$$c_b(\lambda) = 1 + \sum_{j=1}^{r} \frac{|b_j|^2}{\sigma_j^2 - \lambda} - \sum_{j=1}^{2} \frac{|\hat{b}_j|^2}{\lambda} \left(1 + \frac{\hat{\sigma}_j}{\lambda}\right). \tag{72}$$

We used the terms as they are written in (59) because they are available to us in the algorithm, but we could have used the terms from (58) instead. The part that is unique to each function are the components of $e_{f_b}(\lambda)$; Therefore, from (67) we can write

$$u_{g_b}(\lambda) = -\sum_{j=r+1}^{n} \frac{|b_j|^2}{\lambda} \sum_{i=2}^{\infty} \left(\frac{\sigma_j^2}{\lambda}\right)^i, \tag{73}$$

$$u_{f_b}(\lambda) = -\sum_{j=1}^{2} \frac{|\hat{b}_j|^2}{\lambda} \sum_{i=2}^{\infty} \left(\frac{\hat{\sigma}_j}{\lambda}\right)^i. \tag{74}$$

Looking at $u_{g_b}(\lambda)$, we see that it contains an infinite sum over $(\sigma_j^2/\lambda)^i$, which starts at $i = 2$. Because we are only tracking the r largest eigenvalues, we know that $(\sigma_j^2/\lambda) < 1$, and as the SNR goes up, this term will get smaller. There is also an additional $1/\lambda$ term as well which will reduce the unique parts.

Figure 5 shows the difference term $e_{f_b}(\lambda)$, along with $u_{g_b}(\lambda)$ and $u_{g_b}(\lambda)$. Notice that all three curves are around the same magnitude for $\lambda > \sigma_r^2$. This means that $u_{g_b}(\lambda)$, which can easily be calculated in the IFAST algorithm, can be used as an indicator of the error $e_{f_b}(\lambda)$. The plot is in $\log_{10}$ because the differences are so small.

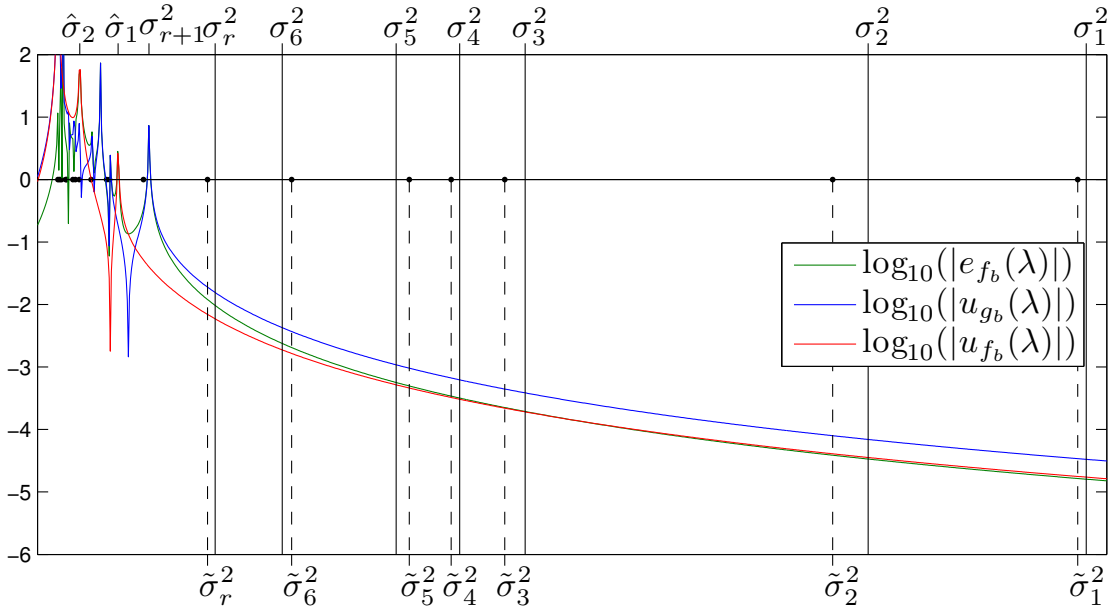


Figure 5. $\log_{10}$ of the difference $e_{f_b}(\lambda) = w_{f_b}(\lambda) - w_{g_b}(\lambda)$ from (55), along with the unique parts of $e_{f_b}(\lambda) = u_{f_b}(\lambda) - u_{g_b}(\lambda)$ from (74) and (73).

1.5 The Rank-Two Case

Looking at (34) and (49), we see that they each consist of three terms. We separated $w_{g_b}(\lambda)$ and $w_{f_b}(\lambda)$ into common and unique parts in the previous section. We will now do the same for $w_{g_a}(\lambda)$ and $w_{f_a}(\lambda)$, as well as $w_{g_x}(\lambda)$ and $w_{f_x}(\lambda)$ in this section.

For $w_{g_a}(\lambda)$ in (34) and $w_{f_a}(\lambda)$ in (49), we can write

$$w_{g_a}(\lambda) = c_a(\lambda) + u_{g_a}(\lambda), \tag{75}$$

$$w_{f_a}(\lambda) = c_a(\lambda) + u_{f_a}(\lambda), \tag{76}$$

where the common part is

$$c_a(\lambda) = 1 - \sum_{j=1}^{r} \frac{|a_j|^2}{\sigma_j^2 - \lambda} + \sum_{j=1}^{2} \frac{|\hat{a}_j|^2}{\lambda} \left(1 + \frac{\hat{\sigma}_j}{\lambda}\right), \tag{77}$$

and the two unique parts are

$$u_{g_a}(\lambda) = \sum_{j=r+1}^{n} \frac{|a_j|^2}{\lambda} \sum_{i=2}^{\infty} \left(\frac{\sigma_j^2}{\lambda}\right)^i, \tag{78}$$

$$u_{f_a}(\lambda) = \sum_{j=1}^{2} \frac{|\hat{a}_j|^2}{\lambda} \sum_{i=2}^{\infty} \left(\frac{\hat{\sigma}_j}{\lambda}\right)^i. \tag{79}$$

For $w_{g_x}(\lambda)$ in (34) and $w_{f_x}(\lambda)$ in (49), we can write

$$w_{g_x}(\lambda) = c_x(\lambda) + u_{g_x}(\lambda), \tag{80}$$

$$w_{f_x}(\lambda) = c_x(\lambda) + u_{f_x}(\lambda), \tag{81}$$

where the common part is

$$c_x(\lambda) = \sum_{j=1}^{r} \frac{a_j^* b_j}{\sigma_j^2 - \lambda} - \sum_{j=1}^{2} \frac{\hat{a}_j^* \hat{b}_j}{\lambda} \left(1 + \frac{\hat{\sigma}_j}{\lambda}\right), \tag{82}$$

and the two unique parts are

$$u_{g_x}(\lambda) = -\sum_{j=r+1}^{n} \frac{a_j^* b_j}{\lambda} \sum_{i=2}^{\infty} \left(\frac{\sigma_j^2}{\lambda}\right)^i, \tag{83}$$

$$u_{f_x}(\lambda) = -\sum_{j=1}^{2} \frac{\hat{a}_j^* \hat{b}_j}{\lambda} \sum_{i=2}^{\infty} \left(\frac{\hat{\sigma}_j}{\lambda}\right)^i. \tag{84}$$

Finally, for the full rank two secular functions, $w_g(\lambda)$ in (33) and $w_f(\lambda)$ in (48), we can write

$$w_g(\lambda) = c(\lambda) + u_g(\lambda), \tag{85}$$

$$w_f(\lambda) = c(\lambda) + u_f(\lambda), \tag{86}$$

where the common part is

$$c(\lambda) = c_a(\lambda) c_b(\lambda) + |c_x(\lambda)|^2, \tag{87}$$

and the two unique parts are

$$u_g(\lambda) = c_a u_{g_b} + c_b u_{g_a} + 2Re\left\{c_x^* u_{g_x}\right\} + u_{g_a} u_{g_b} + |u_{g_x}|^2, \tag{88}$$

$$u_f(\lambda) = c_a u_{f_b} + c_b u_{f_a} + 2Re\left\{c_x^* u_{f_x}\right\} + u_{f_a} u_{f_b} + |u_{f_x}|^2. \tag{89}$$

It is interesting to note that common part of the rank-two secular functions of $w_g(\lambda)$ and $w_f(\lambda)$, from (87), has the same form as the full rank-two secular functions from (34) and (49). All of the terms that differ contain at least one instance of the unique functions, $u(\lambda)$, and we saw from the previous section that these functions are very small in the region where $\lambda > \tilde{\sigma}_r^2$. This means that the offset of the roots of the secular equation, and thus the approximation error, will be small.

Now that we have determined all of the components of $w_g(\lambda)$ and $w_f(\lambda)$, we can write the error in the two-rank two secular function for $\tilde{F}$, (similar to the rank-one version from (55)), as

$$e_f(\lambda) = w_f(\lambda) - w_g(\lambda) = u_f(\lambda) - u_g(\lambda) \tag{90}$$

$$\begin{aligned} e_f(\lambda) &= c_a(\lambda)\left(u_{f_b}(\lambda) - u_{g_b}(\lambda)\right) + c_b(\lambda)\left(u_{f_a}(\lambda) - u_{g_a}(\lambda)\right) \\ &\quad + 2Re\left\{c_x^*(\lambda)\left(u_{f_x}(\lambda) - u_{g_x}(\lambda)\right)\right\} \\ &\quad + u_{f_a}(\lambda)u_{f_b}(\lambda) - u_{g_a}(\lambda)u_{g_b}(\lambda) + |u_{f_x}(\lambda)|^2 - |u_{g_x}(\lambda)|^2. \end{aligned} \tag{91}$$

1.6 Additional Comments

The eigendecomposition of $\tilde{F}$ in Step 3 of the IFAST algorithm tends to dominate the computation for large r. From (44), we can see that any $\tilde{F}$ can be written as the sum of a diagonal matrix, two rank-one matrices and a rank-four matrix which contains all zero values except the lower left $2 \times r$ block, and the upper right $r \times 2$ block. It may be possible to take advantage of this structure by using the method from manuscript 2 to determine the eigendecomposition of $\tilde{F}$ more efficiently.

The inclusion of any reasonable approximation of the missing error term, $e_f(\lambda)$, should only increase the accuracy of the computation, but its impact on the algorithm as a whole would need to be examined. It could lead to a more accurate algorithm.

List of References

[1] E. C. Real, D. W. Tufts, and J. W. Cooley, "Two algorithms for fast approximate subspace tracking," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 47, no. 7, pp. 1036–1045, July 1999.

[2] E. C. Real, R. M. Yannone, and D. W. Tufts, "Comparison of two methods for multispectral 3-D detection of single pixel features in strong textured clutter," in *Proceedings Image and Multidimensional Digital Signal Processing Conference (IMDSP), Alpbach, Austria*, July 1998.

[3] D. W. Tufts, "Keynote address," in *MIT Lincoln Laboratory Adaptive Array Signal Processing (ASAP) Conference, Lexington, MA*, Mar. 2001.

[4] A. A. Shah and D. W. Tufts, "Determination of the dimension of a signal subspace from short data records," *IEEE Transactions on Signal Processing*, vol. 42, no. 9, pp. 2531–2535, Sept. 1994.

[5] D. W. Tufts and A. A. Shah, "Rank determination in time-series analysis," in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP)*, Apr. 1994, pp. IV–21–IV–24.

[6] T. M. Toolan and D. W. Tufts, "Detection and estimation in non-stationary environments," in *Proceedings IEEE Asilomar Conference on Signals, Systems & Computers*, Nov. 2003, pp. 797–801.

[7] I. Karasalo, "Estimating the covariance matrix by signal subspace averaging," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, no. 1, pp. 8–12, Feb. 1986.

[8] G. H. Golub and C. F. van Loan, *Matrix Computations, 3rd ed.* Baltimore, MD: Johns Hopkins Univ. Press, 1996.

[9] J. W. Cooley, P. A. Lewis, and P. D. Welch, "The Fast Fourier Transform algorithm and its applications," IBM Watson Research Center, Yorktown Heights, NY, Tech. Rep. RC-1743, Feb. 1967.

[10] J. W. Cooley and J. W. Tukey, "An algorithm for the machine computation of complex Fourier series," *Mathematics of Computation*, vol. 19, pp. 297–301, Apr. 1965.

[11] J. W. Cooley, T. M. Toolan, and D. W. Tufts, "A subspace tracking algorithm using the Fast Fourier Transform," *IEEE Signal Processing Letters*, vol. 11, no. 1, pp. 30–32, Jan. 2004.

[12] G. H. Golub, "Some modified matrix eigenvalue problems," *SIAM Review*, vol. 15, no. 2, pp. 318–334, Apr. 1973.

[13] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*. London, UK: Oxford Univ. Press, 1965.

[14] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, "Rank-one modification of the symmetric eigenproblem," *Numerische Mathematik*, vol. 31, pp. 31–48, 1978.

MANUSCRIPT 2

Rank-two modification of the symmetric eigenproblem

2.1 Abstract

A method is presented for efficiently computing the eigenvalues and eigenvectors of the sum of a diagonal matrix, D, and two rank one matrices of the form $\tilde{G} = D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H$. The rank-two secular function, the roots of which are the eigenvalues of $\tilde{G}$, is derived, along with a simple way to determine disjoint intervals which bound each root. A non-iterative method to calculate all n of the eigenvectors of $\tilde{G} \in \mathbb{C}^{n\times n}$ in $O(n^2)$ time is also presented.

The above problem is closely related to updating the singular value decomposition (SVD) of a matrix where one row or column has been replaced. A method for determining this update is presented, in which a single $O(n^3)$ matrix product is required to calculate the new right or left singular vectors.

All non-numerical pathological cases are addressed, including creating a duplicate eigenvalue, unchanged eigenvalues, and having duplicate eigenvalues in the original matrix. These pathological cases give insight into higher order perturbations, as well as more numerically stable implementations.

2.2 Introduction

Using the secular equation to update the eigenvalues of a rank-one modification to a diagonal matrix, as in $D - \boldsymbol{a}\boldsymbol{a}^H$, has been analyzed by Wilkinson [1] and Golub [2]. A non-iterative method to determine the eigenvectors, along with a more numerically robust method to determine eigenvalues was described by Bunch, Nielsen, and Sorensen [3]. Analysis and improvement of the numerical properties, especially of the eigenvectors, has been addressed more recently by Gu and Eisenstat [4, 5]. One important property of this method is that it can determine both

the eigenvalues and eigenvectors of a rank-one modification to a diagonal matrix in $O(n^2)$ time. This allows one to update the eigendecomposition of a rank-one modification to any symmetric matrix with known eigendecomposition with only a single $O(n^3)$ matrix product.

In this manuscript, we present a method to determine the eigenvalues and eigenvectors of two simultaneous rank-one modifications to a diagonal matrix, as in $D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H$, in $O(n^2)$ time. This allows one to update the eigendecomposition of a *rank-two* modification to any symmetric matrix with known eigendecomposition with only a single $O(n^3)$ matrix product, just like in the rank-one case. A common place where a problem of this form appears is when we have a matrix where we simultaneously remove and add a column (or modify a single column), and want to update the singular value decomposition of the new matrix.

As an algorithm, the method presented in this manuscript can be more efficient in updating an eigendecomposition than using the rank-one method twice because the $O(n^3)$ rotation of the eigenvectors is the dominant computation. As an analysis tool, having a formula which directly calculates the eigenvalues of the rank-two modification can be more revealing than two sequential rank-one calculations with a rotation in the middle.

The organization of this manuscript is as follows. Section 2.3 introduces the basic notation, and provides an example of the applicability of this method. Section 2.4 shows how to deflate the problem to exclude eigenvalues whose eigenvectors don't change, and to exclude some multiple eigenvalues. Section 2.5 derives the rank-two secular function, the roots of which are the eigenvalues of $D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H$, and shows how to find disjoint intervals which bound each root. Section 2.6 shows how to determine the eigenvectors using a non-iterative method. Section 2.7 gives a summary of the necessary steps required to perform the method described in this

manuscript.

2.3 Defining the Problem

Our new formulas and method for computing the eigenvalues and eigenvectors of a rank-two modification of a diagonal matrix have wider applicability than one might infer from the introduction. As long as we can convert a given problem into a rank-two modification of a diagonal matrix of the form $D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H$, the method is applicable.

In this section, we show that the problem of updating the SVD of a rectangular matrix where one column has been replaced can be recast into the form of $D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H$. We start with the matrix $M \in \mathbb{C}^{r\times c}$, along with its SVD,

$$M = U_M \Sigma_M V_M^H. \tag{92}$$

If we define $n = \min(r, c)$ to be the smaller dimension of M, then $U_M \in \mathbb{C}^{r\times n}$ and $V_M \in \mathbb{C}^{c\times n}$ are matrices with orthogonal and normal columns, where $U_M^H U_M = I$ and $V_M^H V_M = I$. The matrix $\Sigma_M \in \mathbb{R}^{n\times n}$ is diagonal, with descending non-negative values on the diagonal. We can write M as c, length r, column vectors,

$$M = \begin{bmatrix} \boldsymbol{m}_1 & \boldsymbol{m}_2 & \boldsymbol{m}_3 & \cdots & \boldsymbol{m}_{c-1} & \boldsymbol{m}_c \end{bmatrix}. \tag{93}$$

We then create the matrix $\tilde{M} \in \mathbb{C}^{r\times c}$, by shifting the $c-1$ rightmost columns of M one column to the left, discarding the leftmost column, and adding a new column on the right,

$$\tilde{M} = \begin{bmatrix} \boldsymbol{m}_2 & \boldsymbol{m}_3 & \cdots & \boldsymbol{m}_{c-1} & \boldsymbol{m}_c & \boldsymbol{m}_{c+1} \end{bmatrix}. \tag{94}$$

We would like to calculate the SVD of $\tilde{M}$ by taking advantage of the fact that it shares all but one column with M, whose SVD we already have.

We can write $\tilde{M}$ as

$$\tilde{M} = \left(M - \boldsymbol{m}_1 \boldsymbol{e}_1^T\right) P + \boldsymbol{m}_{c+1} \boldsymbol{e}_c^T \tag{95}$$

where $\boldsymbol{e}_1$ and $\boldsymbol{e}_c$ are the 1st and cth length c canonical vectors, respectively, and P is a $c \times c$ matrix of these length c canonical vectors,

$$\boldsymbol{e}_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \qquad \boldsymbol{e}_c = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \qquad P = \left[\begin{array}{cccc|c} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{array}\right]. \tag{96}$$

Note that the ith length c canonical vector, $\boldsymbol{e}_i$, is the ith column of a $c \times c$ identity matrix. Multiplying $\tilde{M}$ as defined in (95) by its conjugate transpose, we get

$$\tilde{M}\tilde{M}^H = MM^H - \boldsymbol{m}_1\boldsymbol{m}_1^H + \boldsymbol{m}_{c+1}\boldsymbol{m}_{c+1}^H. \tag{97}$$

Multiplying (97) by U_M^H on the left and U_M on the right, we get

$$U_M^H\tilde{M}\tilde{M}^H U_M = \Sigma_M^2 - U_M^H\boldsymbol{m}_1\boldsymbol{m}_1^H U_M + U_M^H\boldsymbol{m}_{c+1}\boldsymbol{m}_{c+1}^H U_M, \tag{98}$$

where the Σ_M^2 term comes from $U_M^H MM^H U_M$. Next, we define the diagonal matrix $D \in \mathbb{R}^{n\times n}$, and the column vectors $\boldsymbol{a} \in \mathbb{C}^n$ and $\boldsymbol{b} \in \mathbb{C}^n$ as

$$D = \Sigma_M^2, \qquad \boldsymbol{a} = U_M^H\boldsymbol{m}_1, \qquad \boldsymbol{b} = U_M^H\boldsymbol{m}_{c+1}. \tag{99}$$

Thus, $\boldsymbol{a}$ is a rotated version of the column we are discarding, and $\boldsymbol{b}$ is a rotated version of the column we are adding. The ith element on the diagonal of D will be d_i, and the ith element of $\boldsymbol{a}$ and $\boldsymbol{b}$ will be $a_i = \boldsymbol{u}_{i,M}^H\boldsymbol{m}_1$ and $b_i = \boldsymbol{u}_{i,M}^H\boldsymbol{m}_{c+1}$, respectively, where $\boldsymbol{u}_{i,M}$ is the ith left singular vector of M, which is the ith column of U_M. If we write

$$\tilde{G} = D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H, \tag{100}$$

it is obvious that $\tilde{G}$ is a diagonal matrix plus two rank-one matrices. From (98), we see $\tilde{G} = U_M^H\tilde{M}\tilde{M}^H U_M$. Writing the eigendecomposition of $\tilde{G}$ as

$$\tilde{G} = \tilde{U}\tilde{D}\tilde{U}^H, \tag{101}$$

then multiplying (101) by U_M and U_M^H, we get

$$\tilde{M}\tilde{M}^H = U_M\tilde{U}\tilde{D}\tilde{U}^H U_M^H = (U_M\tilde{U})\tilde{D}(U_M\tilde{U})^H. \tag{102}$$

This means the eigenvalues of $\tilde{G}$ are the squares of the singular values of $\tilde{M}$, and the eigenvectors of $\tilde{G}$, rotated by U_M, are the left singular vectors of $\tilde{M}$. Thus the singular values and left singular vectors of $\tilde{M}$ can be written as

$$\tilde{\Sigma}_M = \sqrt{\tilde{D}}, \qquad \tilde{U}_M = U_M\tilde{U}. \tag{103}$$

The right singular vectors of $\tilde{M}$ can be calculated as $\tilde{V}_M = \tilde{M}^H\tilde{U}_M\tilde{\Sigma}_M^{-1}$. Thus, we can update the SVD of $\tilde{M}$ by determining the eigendecomposition of $\tilde{G}$.

2.4 Sorting and Deflating

Before we start calculating the eigendecomposition of $\tilde{G}$, we must first sort the elements on the diagonal of D in ascending order (which is the reverse of the SVD), deflate any duplicate eigenvalues to a multiplicity of at most two, and remove any eigenvalues whose corresponding eigenvectors don't change.

2.4.1 Sorting D

The old eigenvalues, which are the elements on the diagonal of D, must be sorted in ascending order to determine the bounds for the new eigenvalues. This can be done by creating a permutation matrix, I_s, which is just an $n \times n$ identity matrix with the columns reordered in the same order as required to sort the elements on the diagonal of D. If D is already sorted in ascending order, I_s will be the identity matrix, if D is sorted in descending order, as with the SVD, I_s will be an anti-diagonal matrix with ones on the anti-diagonal. If we then define $D' = I_s^H D I_s$, $\boldsymbol{a}' = I_s^H\boldsymbol{a}$, and $\boldsymbol{b}' = I_s^H\boldsymbol{b}$, then D' will have its diagonal sorted in ascending order. If we then determine the eigendecomposition of the sorted matrix

$$\tilde{G}' = D' - \boldsymbol{a}'\boldsymbol{a}'^H + \boldsymbol{b}'\boldsymbol{b}'^H = \tilde{U}'\tilde{D}'\tilde{U}'^H, \tag{104}$$

the eigendecomposition of $\tilde{G}$ can be written as $\tilde{D} = \tilde{D}'$, and $\tilde{U} = I_s\tilde{U}'$. Note that multiplying by I_s from the left is just a reordering of rows, and multiplying by I_s from the right is just a reordering of columns, and neither actually requires any computation.

2.4.2 Deflating $\tilde{G}$

If $a_i = b_i = 0$ for a given value of i, then neither the ith eigenvalue nor the ith eigenvector will change, therefore we can remove these elements from $\tilde{G}$. Before we do that, we will rotate the eigenvectors that correspond to eigenvalues with a multiplicity greater than one so that we can produce more elements of $\boldsymbol{a}$ and $\boldsymbol{b}$ that meet the condition $a_i = b_i = 0$.

Rotating multiple eigenvalues

If there are any eigenvalues with multiplicity greater than two, (including multiple zeros), we must reduce them to a multiplicity of at most two. Whenever a matrix has an eigenvalue of multiplicity k, where $k > 1$, the corresponding eigenvectors are confined to a k dimensional subspace of arbitrary rotation. This means that any rotation of those k eigenvectors by a unitary matrix will still be eigenvectors of that matrix. If there are k identical eigenvalues at d_i through d_{i+k-1}, and we define the matrix $Y \in \mathbb{C}^{k\times 2}$ to contain the elements from $\boldsymbol{a}$ and $\boldsymbol{b}$ which correspond to the duplicate eigenvalue,

$$Y = \begin{bmatrix} a_i & a_{i+1} & a_{i+2} & \cdots & a_{i+k-1} \\ b_i & b_{i+1} & b_{i+2} & \cdots & b_{i+k-1} \end{bmatrix}^T, \tag{105}$$

and take its QR factorization as $Y = QR$, then Q will be a unitary matrix, and R will be a $k \times 2$ upper triangular matrix with at most three nonzero elements.

To illustrate how we take advantage of this, we define the matrix I_Y to be an $n \times n$ identity matrix with the $k \times k$ matrix Q inserted on the diagonal in the location of the duplicated eigenvalue. This block will be at the intersection of the

ith through $(i+k)$th rows and columns of I_Y. We can then rewrite (100) as

$$\tilde{G} = I_Y \left(D - I_Y^H \boldsymbol{a}\boldsymbol{a}^H I_Y + I_Y^H \boldsymbol{b}\boldsymbol{b}^H I_Y \right) I_Y^H. \tag{106}$$

The reason the D term does not change is because $D = I_Y^H D I_Y$. The term $I_Y^H \boldsymbol{a}$ is the same as $\boldsymbol{a}$ in all elements except the ith through the $(i+k)$th, where it equals the first column of R, and similarly $I_Y^H \boldsymbol{b}$ is the same as $\boldsymbol{b}$ in all elements except the ith through the $(i+k)$th, where it equals the second column of R,

$$\begin{aligned} I_Y^H \boldsymbol{a} &= \left[\begin{array}{ccc|cccc|ccc} a_1 & \cdots & a_{i-1} & r_{1,1} & 0 & 0 & \cdots & 0 & a_{i+k} & \cdots & a_n \end{array} \right]^T \\ I_Y^H \boldsymbol{b} &= \left[\begin{array}{ccc|cccc|ccc} b_1 & \cdots & b_{i-1} & r_{1,2} & r_{2,2} & 0 & \cdots & 0 & b_{i+k} & \cdots & b_n \end{array} \right]^T. \end{aligned} \tag{107}$$

This means that if we calculate the eigendecomposition of the expression inside the parenthesis of (106), we will have inserted at least $k-2$ elements into $\boldsymbol{a}$ and $\boldsymbol{b}$ where $a_i = b_i = 0$. If the two columns of Y are linearly dependent, $r_{2,2}$ will be zero, and we will be able to remove an additional eigenvalue.

If all of a_i through a_{i+k-1} are zero, $r_{1,1}$ will be zero, and we can deflate all but one of the identical eigenvalues in this case.

To apply this for all eigenvalues with multiplicity of two or more, we start with an $n \times n$ identity matrix, which we will call I_r. Find all eigenvalues on the diagonal of D' (the sorted version of D) with a multiplicity greater than or equal to two, calculate the QR factorization of the portion of $\boldsymbol{a}'$ and $\boldsymbol{b}'$ which corresponds to the duplicate eigenvalue, and insert this Q into I_r on the diagonal as was done in the above example with I_Y. If we define $\dot{\boldsymbol{a}} = I_r^H \boldsymbol{a}'$ and $\dot{\boldsymbol{b}} = I_r^H \boldsymbol{b}'$, then $\dot{\boldsymbol{a}}$ and $\dot{\boldsymbol{b}}$ will be $\boldsymbol{a}'$ and $\boldsymbol{b}'$ with values from the R matrices of the QR factorizations inserted in a similar fashion as (107).

Removing the unchanged eigenvalues

Now that all of the rotations are done, we have the matrix $I_r \in \mathbb{C}^{n \times n}$, which if there were no duplicate eigenvalues is just the identity matrix, and if there were

duplicate eigenvalues, will have some Q blocks on the diagonal. The final step is to remove the unchanged eigenvalues.

We start with a matrix I_d, which we will initially set to I_r. Using $\dot{\boldsymbol{a}}$ and $\dot{\boldsymbol{b}}$, find each i where $\dot{a}_i = \dot{b}_i = 0$. For each of these values of i, remove the ith column from I_d, and place it in a matrix called $\tilde{U}_d$. If n_d is the number of values that satisfy $\dot{a}_i = \dot{b}_i = 0$, we will end up with the two matrices $I_d \in \mathbb{C}^{n \times n - n_d}$ and $\tilde{U}_d \in \mathbb{C}^{n \times n_d}$. We can then write the deflated terms as

$$D'' = I_d^H I_s^H D I_s I_d, \qquad \boldsymbol{a}'' = I_d^H I_s^H \boldsymbol{a}, \qquad \boldsymbol{b}'' = I_d^H I_s^H \boldsymbol{b}. \tag{108}$$

If we determine the eigendecomposition of the deflated matrix

$$\tilde{G}'' = D'' - \boldsymbol{a}''\boldsymbol{a}''^H + \boldsymbol{b}''\boldsymbol{b}''^H = \tilde{U}''\tilde{D}''\tilde{U}''^H, \tag{109}$$

then the eigendecomposition of $\tilde{G}$ can be written as

$$\tilde{D} = \left[\begin{array}{c|c} \tilde{D}_d & 0 \\ \hline 0 & \tilde{D}'' \end{array}\right], \qquad \tilde{U} = I_s \left[\begin{array}{c|c} \tilde{U}_d & I_d\tilde{U}'' \end{array}\right], \tag{110}$$

where $\tilde{D}_d = \tilde{U}_d^H I_s^H D I_s \tilde{U}_d$ is the $n_d \times n_d$ matrix of eigenvalues that did not change.

2.4.3 Replacing $\tilde{G}$ with the deflated version

It is important to note that there is almost no computation required to do the actual deflation. Although this section has been written using many matrix products, in practice almost none of them actually need be performed. The only computations required are calculating the QR factorization of a $k \times 2$ matrix for each duplicate eigenvalue, which even for large values of k is negligible, and the product $I_d\tilde{U}''$, which is also negligible because I_d is mostly columns from an identity matrix, with small blocks that contain at most two columns of Q from each QR factorization.

From this point on in the manuscript, we will assume $\tilde{G}$ has been sorted and deflated as described in this section, and when we refer to $\tilde{G}$, D, $\boldsymbol{a}$, $\boldsymbol{b}$, and n, we really mean the sorted and deflated versions, $\tilde{G}''$, D'', $\boldsymbol{a}''$, $\boldsymbol{b}''$, and $(n - n_d)$.

2.5 The eigenvalues of $\tilde{G}$

In this section, we will derive the equivalent of the secular equation [2] for $\tilde{G}$, which we will call the *rank-two secular equation*, then show how to bound each root in disjoint intervals which can be used to calculate the eigenvalues of $\tilde{G}$. The secular equation has the same roots as the characteristic equation, but has some better properties for finding the roots [1, 2].

2.5.1 The characteristic polynomial of $\tilde{G}$

The characteristic polynomial of a matrix, whose roots are the eigenvalues of that matrix, is the determinant of that matrix minus λ times the identity matrix. The characteristic polynomial of $\tilde{G}$ is

$$C(\lambda) = \det\left[D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H - \lambda I\right]. \tag{111}$$

We will define the diagonal matrix

$$T_\lambda = (D - \lambda I), \tag{112}$$

whose ith diagonal element is $t_i = (d_i - \lambda)$. Expanding (111) in terms of t_i and the elements of $\boldsymbol{a}$ and $\boldsymbol{b}$, we get

$$C(\lambda) = \det\begin{bmatrix} b_1 b_1^* - a_1 a_1^* + t_1 & b_1 b_2^* - a_1 a_2^* & \cdots & b_1 b_n^* - a_1 a_n^* \\ b_2 b_1^* - a_2 a_1^* & b_2 b_2^* - a_2 a_2^* + t_2 & \cdots & b_2 b_n^* - a_2 a_n^* \\ \vdots & \vdots & \ddots & \vdots \\ b_n b_1^* - a_n a_1^* & b_n b_2^* - a_n a_2^* & \cdots & b_n b_n^* - a_n a_n^* + t_n \end{bmatrix}. \tag{113}$$

Using the determinant properties that:

- a scalar can be factored out of any row or column, and
- any two rows or columns can be added together without changing the determinant,

we can rewrite (113) as the determinant of a matrix which has nonzero values only on the diagonal and the first two rows and columns, and simple values everywhere except the upper left 2×2 block. To do this, we start by setting W equal to $C(\lambda)$ as it appears in (113), and $p = 1$, which will be used to collect the row and column scale factors. Thus, we can write

$$C(\lambda) = pW. \tag{114}$$

We next apply the above two determinant properties to W and p successively as shown in Table 5 to change the form of $C(\lambda)$ to that of (115). The notation used for the rows and columns of W in Table 5 are $W_{i,:}$ for the ith row of W, and $W_{:,i}$ for the ith column of W.

$$C(\lambda) = t_1 t_2 \det \begin{bmatrix} \frac{a_1^* b_1}{t_1} + \frac{a_2^* b_2}{t_2} & \frac{a_1 a_1^*}{t_1} + \frac{a_2 a_2^*}{t_2} - 1 & a_3^* & a_4^* & a_5^* & \cdots & a_n^* \\ \frac{b_1 b_1^*}{t_1} + \frac{b_2 b_2^*}{t_2} + 1 & \frac{a_1 b_1^*}{t_1} + \frac{a_2 b_2^*}{t_2} & b_3^* & b_4^* & b_5^* & \cdots & b_n^* \\ -b_3 & -a_3 & t_3 & 0 & 0 & \cdots & 0 \\ -b_4 & -a_4 & 0 & t_4 & 0 & \cdots & 0 \\ -b_5 & -a_5 & 0 & 0 & t_5 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -b_n & -a_n & 0 & 0 & 0 & \cdots & t_n \end{bmatrix}. \tag{115}$$

Now that the determinant portion of $C(\lambda)$ is primarily zeros, we can evaluate it much more efficiently. Looking ahead, section 2.9 gives a closed form for the determinant of a matrix of the form of (115). Plugging the elements from (115) into (160), re-substituting $t_i = (d_i - \lambda)$, and doing some algebraic manipulation, we get a closed form for the characteristic polynomial of $\tilde{G}$,

$$\begin{aligned} C(\lambda) = \left(\prod_{i=1}^{n} (d_i - \lambda) \right) \Bigg(1 - \sum_{j=1}^{n} \frac{|a_j|^2}{d_j - \lambda} + \sum_{j=1}^{n} \frac{|b_j|^2}{d_j - \lambda} \\ - \left(\sum_{j=1}^{n} \frac{|a_j|^2}{d_j - \lambda} \right) \left(\sum_{l=1}^{n} \frac{|b_l|^2}{d_l - \lambda} \right) + \left(\sum_{j=1}^{n} \frac{a_j b_j^*}{d_j - \lambda} \right) \left(\sum_{l=1}^{n} \frac{a_l^* b_l}{d_l - \lambda} \right) \Bigg), \end{aligned} \tag{116}$$

Table 5. The steps to convert the determinant portion of $C = pW$ to a bordered diagonal matrix.

Step	Modifications to p	Modifications to W	Range for i
1)	$p = pb_i^*$	$W_{:,i} = W_{:,i}/b_i^*$	$i = 1 \cdots n$
2)		$W_{:,i} = W_{:,i} - W_{:,1}$	$i = 2 \cdots n$
3)		$W_{:,i} = W_{:,i} - W_{:,2}$	$i = 3 \cdots n$
4)	$p = pa_i$	$W_{i,:} = W_{i,:}/a_i$	$i = 1 \cdots n$
5)†	$p = pW_{1,i}$	$W_{:,i} = W_{:,i}/W_{1,i}$	$i = 3 \cdots n$
6)		$W_{i,:} = W_{i,:} - W_{1,:}$	$i = 2 \cdots n$
7)	$p = p/(a_i^* - a_2^* b_i^*/b_2^*)$	$W_{:,i} = W_{:,i}(a_i^* - a_2^* b_i^*/b_2^*)$	$i = 3 \cdots n$
8)	$p = -p/(1 + b_1 b_1^*/t_1)$	$W_{:,2} = -W_{:,2}(1 + b_1 b_1^*/t_1)$	
9)		$W_{:,1} = W_{:,1} - W_{:,2}$	
10)	$p = -p(t_1/b_1^* + b_1)/a_1$	$W_{:,2} = -W_{:,2} a_1/(t_1/b_1^* + b_1)$	
11)	$p = -pt_2/(a_2 a_2^*)$	$W_{2,:} = -W_{2,:} a_2 a_2^*/t_2$	
12)		$W_{1,:} = W_{1,:} - W_{2,:}$	
13)	$p = -p/a_i$	$W_{i,:} = -W_{i,:} a_i$	$i = 3 \cdots n$
14)	$p = -pa_2^*/b_2^*$	$W_{2,:} = -W_{2,:} b_2^*/a_2^*$	

† p must be calculated before $W_{1,i}$ is modified

which when factored gives us

$$C(\lambda) = \left(\prod_{i=1}^{n}(d_i - \lambda)\right)\left(\left(1 - \sum_{j=1}^{n}\frac{|a_j|^2}{d_j - \lambda}\right)\left(1 + \sum_{j=1}^{n}\frac{|b_j|^2}{d_j - \lambda}\right) + \left|\sum_{j=1}^{n}\frac{a_j b_j^*}{d_j - \lambda}\right|^2\right). \tag{117}$$

2.5.2 The rank-two secular function

Since we are only interested in the roots of the characteristic polynomial, we can divide both sides of (117) by $\prod_{i=1}^{n}(d_i - \lambda)$ to get the rank-two secular function,

$$w(\lambda) = \left(1 - \sum_{j=1}^{n}\frac{|a_j|^2}{d_j - \lambda}\right)\left(1 + \sum_{j=1}^{n}\frac{|b_j|^2}{d_j - \lambda}\right) + \left|\sum_{j=1}^{n}\frac{a_j b_j^*}{d_j - \lambda}\right|^2. \tag{118}$$

This function will have the same roots as $C(\lambda)$, but will have poles at the original eigenvalues, which are the elements on the diagonal of D.

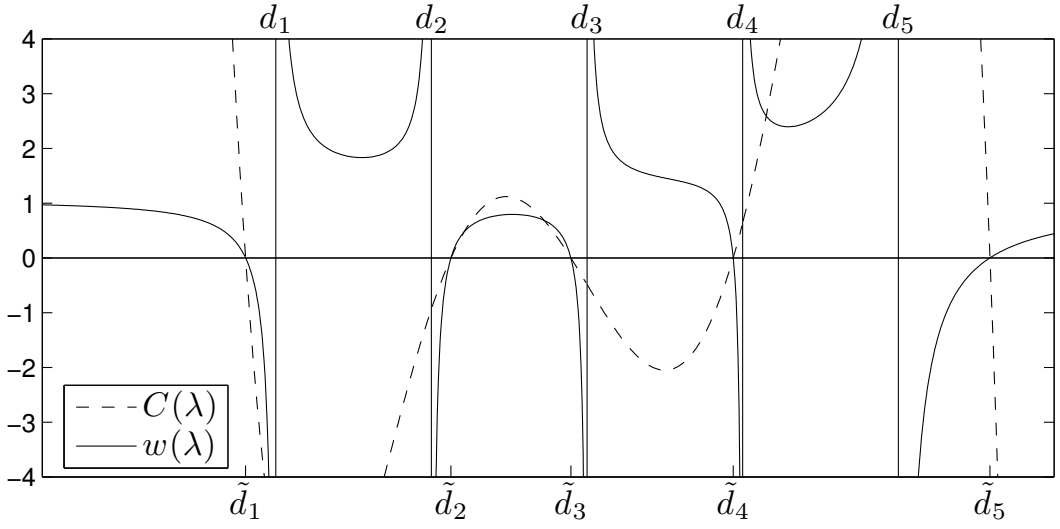


Figure 6. A simple example to illustrate the rank-two secular function and characteristic polynomial of $\tilde{G} = D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H$. The values used to create this figure were $D = \text{diag}(d_1, d_2, d_3, d_4, d_5) = \text{diag}(1, 2, 3, 4, 5)$, $\boldsymbol{a} = [0.8, 0.26, 0.4, 0.53, 0.67]^T$, and $\boldsymbol{b} = [0.69, 0.46, 0.23, 0.46, 1]^T$. The eigenvalues of $\tilde{G}$ are $\tilde{D} = \text{diag}(\tilde{d}_1, \tilde{d}_2, \tilde{d}_3, \tilde{d}_4, \tilde{d}_5) \simeq \text{diag}(0.81, 2.13, 2.9, 3.94, 5.59)$.

Figure 6 shows an example of the rank-two secular function and the characteristic polynomial for a 5×5 matrix. Unlike the rank-one secular function, which can have only one root in the interval between adjacent eigenvalues of D, the rank-two secular function can have zero, one, or two roots between adjacent old eigenvalues.

If we define the three terms,

$$w_a(\lambda) = 1 - \sum_{j=1}^{n} \frac{|a_j|^2}{d_j - \lambda}, \qquad w_b(\lambda) = 1 + \sum_{j=1}^{n} \frac{|b_j|^2}{d_j - \lambda}, \qquad w_{ab}(\lambda) = \sum_{j=1}^{n} \frac{a_j^* b_j}{d_j - \lambda}, \tag{119}$$

where $w_a(\lambda)$ is the secular function for $D - \boldsymbol{a}\boldsymbol{a}^H$, and $w_b(\lambda)$ is the secular function for $D + \boldsymbol{b}\boldsymbol{b}^H$ [1], we can write the rank-two secular function as

$$w(\lambda) = w_a(\lambda) w_b(\lambda) + |w_{ab}(\lambda)|^2. \tag{120}$$

From (120) it can be seen that the rank-two secular function is the product of two rank-one secular functions plus a positive cross term. The reader can substitute

special cases, such as $\boldsymbol{a} = \mathbf{0}$, or the case when $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly dependent, to show that (118) will reduce to the case of a rank-one modification of D. Using the diagonal matrix T_λ from (112), we can write (119) in matrix form as

$$w_a(\lambda) = 1 - \boldsymbol{a}^H T_\lambda^{-1} \boldsymbol{a}, \qquad w_b(\lambda) = 1 + \boldsymbol{b}^H T_\lambda^{-1} \boldsymbol{b}, \qquad w_{ab}(\lambda) = \boldsymbol{a}^H T_\lambda^{-1} \boldsymbol{b}. \quad (121)$$

2.5.3 Bounding the roots of $w(\lambda)$

Now that we have the rank-two secular function of $\tilde{G}$, we would like to determine disjoint intervals that bound each root of $w(\lambda)$.

Determining overlapping intervals

We have, by the eigenvalue *separation theorem* [1] (p. 96), bounds for the case of a rank-one perturbation of a matrix. Therefore we will apply that theorem twice to get overlapping intervals for the rank-two case.

We first look at the effect of subtracting the rank-one matrix $\boldsymbol{a}\boldsymbol{a}^H$ from D. If we define a matrix $\dot{G} = D - \boldsymbol{a}\boldsymbol{a}^H$, then by the separation theorem, the eigenvalues of $\dot{G}$, which we will call $\dot{d}_i$, are bounded by

$$d_0 \le \dot{d}_1 \le d_1 \le \dot{d}_2 \le d_2 \le \cdots \le d_{n-1} \le \dot{d}_n \le d_n, \quad (122)$$

where $d_0 = d_1 - \|\boldsymbol{a}\|_2^2$ is the lower bound. Next, we will look at the effect of adding $\boldsymbol{b}\boldsymbol{b}^H$ to $\dot{G}$. We know that $\tilde{G} = \dot{G} + \boldsymbol{b}\boldsymbol{b}^H$; therefore, the eigenvalues of $\tilde{G}$ are bounded by

$$\dot{d}_1 \le \tilde{d}_1 \le \dot{d}_2 \le \tilde{d}_2 \le \dot{d}_3 \le \cdots \le \dot{d}_n \le \tilde{d}_n \le \dot{d}_{n+1}, \quad (123)$$

where $\dot{d}_{n+1} = \dot{d}_n + \|\boldsymbol{b}\|_2^2$ is the upper bound.

Combining (122) and (123), we get overlapping intervals for the eigenvalues of $\tilde{G}$ with respect to the eigenvalues of D,

$$d_{i-1} \le \tilde{d}_i \le d_{i+1}, \qquad i = 1, \cdots, n, \quad (124)$$

where $d_0 = d_1 - \|\boldsymbol{a}\|_2^2$ and $d_{n+1} = d_n + \|\boldsymbol{b}\|_2^2$.

Creating open intervals

We will now show that the intervals in (124) are really open intervals; thus, if the characteristic polynomial, $C(\lambda)$, equals zero when evaluated at the ith old eigenvalue, it will be a single root, and the ith new eigenvalue will equal the ith old eigenvalue.

The only way that the smallest eigenvalue can meet the lower bound of $\tilde{d}_1 = d_0$, is if all but the first element of $\boldsymbol{a}$ are zero, while the first element of $\boldsymbol{b}$ is zero. If this is the case, the eigendecomposition of $D - \boldsymbol{a}\boldsymbol{a}^H$ is exactly the same as the eigendecomposition of D except the smallest eigenvalue is now $d_1 - \|\boldsymbol{a}\|_2^2$. This means we can modify d_1 to be $d_1 - \|\boldsymbol{a}\|_2^2$, and calculate the rank one modification of $D + \boldsymbol{b}\boldsymbol{b}^H$. Thus, we can ignore the equality in $d_0 \leq \tilde{d}_1$ in (124). Similarly, the only way that the largest eigenvalue can meet the upper bound of $\tilde{d}_n = d_{n+1}$, is if all except the nth element of $\boldsymbol{b}$ are zero, while the nth element of $\boldsymbol{a}$ is zero. If this is the case, the eigendecomposition of $D + \boldsymbol{b}\boldsymbol{b}^H$ is exactly the same as the eigendecomposition of D, except that the largest eigenvalue is now $d_n + \|\boldsymbol{b}\|_2^2$. Therefore, we can ignore the equality in $\tilde{d}_n \leq d_{n+1}$ in (124).

When the ith and $(i+1)$th eigenvalues are the same, the characteristic polynomial, $C(\lambda)$, evaluated at the duplicate eigenvalue simplifies to

$$C(d_i) = \left(\prod_{\substack{j=1 \\ j \neq i, i+1}}^{n} (d_j - d_i) \right) \left(- |a_i|^2 |b_{i+1}|^2 - |a_{i+1}|^2 |b_i|^2 + 2\mathrm{Re}\{ a_i b_i^* a_{i+1}^* b_{i+1} \} \right). \tag{125}$$

Because a_i, b_i, a_{i+1}, and b_{i+1} come from the 2×2 upper triangular part of the matrix R from a QR factorization, a_i and b_{i+1} are nonzero, whereas a_{i+1} is zero; therefore, both $|a_{i+1}|^2 |b_i|^2$ and $\mathrm{Re}\{a_i b_i^* a_{i+1}^* b_{i+1}\}$ from (125) are zero. Because the eigenvalues are sorted, the sign of $(d_j - d_i)$ is just the sign of $(j - i)$; therefore, the sign of the product over j in the beginning of (125) simplifies to $(-1)^{i-1}$, giving

us an expression for the sign of $C(\lambda)$ at the duplicate eigenvalue as

$$s_d(i) = (-1)^i. \tag{126}$$

Because (126) cannot equal zero, both eigenvalues at $\lambda = d_i$ must change, and because (126) is only a function of i, this tells us that one of the eigenvalues must increase, while the other decreases.

If all of the elements of $\boldsymbol{a}$ are nonzero, we can replace all of the "$\leq$" relations in (122) with "$<$" [1] (p. 96, eq. 39.8). When (122) is combined with (123), there will be no equality in the bounds. We can make a similar argument with $\boldsymbol{b}$.

When the ith element of $\boldsymbol{a}$ is zero, we get the two bounds, $d_{i-1} < \dot{d}_{i+1} < d_{i+1}$ and $d_{i-1} < \dot{d}_i = d_i < d_{i+1}$, which when combined and sorted can be written as

$$d_{i-1} < \dot{d}_i \leq d_i \leq \dot{d}_{i+1} < d_{i+1}. \tag{127}$$

Equation (127) shows why the equality is needed in (122). What we mean by sorting, is that before sorting, $\dot{d}_{i+1}$ can be less than, greater than, or equal to d_i, but if $\dot{d}_{i+1} < d_i$ we will swap both $\dot{d}_{i+1}$ with $\dot{d}_i$ as well as $\dot{\boldsymbol{u}}_{i+1}$ with $\dot{\boldsymbol{u}}_i$, where $\dot{\boldsymbol{u}}_i$ is the eigenvector of $D - \boldsymbol{a}\boldsymbol{a}^H$ corresponding to the eigenvalue $\dot{d}_i$. The problem with (127) is that it does not indicate that either $\dot{d}_i$ or $\dot{d}_{i+1}$ must equal d_i, while the other may or may not equal d_i, which is actually the case.

Because of deflation, if a_i equals zero, then b_i cannot be zero. This means that if $\dot{d}_i = d_i = \dot{d}_{i+1}$, then one of the eigenvalues at d_i will increase, while the other will remain the same, but no other eigenvalues can move past or onto d_i. If only one of $\dot{d}_i$ or $\dot{d}_{i+1}$ equals d_i, then the eigenvalue at d_i will increase, and no other eigenvalues can move past or onto d_i.

What all of this means, is that there can only be a single root of $C(\lambda)$ at d_i, and it must be the ith root of $C(\lambda)$. This allows us to write (124) as

$$d_{i-1} < \tilde{d}_i < d_{i+1}, \qquad i = 1, \cdots, n. \tag{128}$$

Separating the intervals

From (128), we can see that each eigenvalue is bounded by the next smaller and next larger corresponding old eigenvalue. We will now divide each interval by determining if the new eigenvalue, $\tilde{d}_i$, is greater than, less than, or equal to the corresponding old eigenvalue, d_i.

When $C(\lambda)$ is evaluated at the non duplicate old eigenvalues, it simplifies to

$$C(d_i) = \left(\prod_{\substack{j=1 \\ j \neq i}}^{n} (d_j - d_i) \right) \left(|b_i|^2 - |a_i|^2 - \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{|a_i b_j - a_j b_i|^2}{(d_j - d_i)} \right). \tag{129}$$

Because the eigenvalues are sorted, the sign of $(d_j - d_i)$ is just the sign of $(j - i)$. Therefore, the sign of the product over j in the beginning of (129) simplifies to $(-1)^{i-1}$. This gives us an expression for the sign of $C(d_i)$,

$$s_e(i) = (-1)^{i-1} \text{sgn} \left(|b_i|^2 - |a_i|^2 - \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{|a_i b_j - a_j b_i|^2}{(d_j - d_i)} \right), \tag{130}$$

where *sgn* is the signum function. If $s_e(i)$ evaluates to zero, that means the ith eigenvalue will not change, and $\tilde{d}_i = d_i$.

Because $\tilde{d}_i$ is the only new eigenvalue that can be either greater than *or* less than d_i, and since $\tilde{d}_i$ is the ith root of $C(\lambda)$, we can say that if the sign of $C(\lambda)$ evaluated at d_i (the ith old eigenvalue) is the same as the sign of the slope of $C(\lambda)$ evaluated at $\tilde{d}_i$ (the ith root of $C(\lambda)$), then $\tilde{d}_i$ is less than d_i, otherwise it is greater than d_i. In other words, if the slope of $C(\lambda)$ at $\tilde{d}_i$ is positive, then because $C(\lambda)$ has no roots between $\tilde{d}_i$ and d_i, $C(d_i)$ must be less than zero if $\tilde{d}_i > d_i$, and $C(d_i)$ must be greater than zero if $\tilde{d}_i < d_i$. Alternately, if the slope of $C(\lambda)$ at $\tilde{d}_i$ is negative, $C(d_i)$ must be greater than zero if $\tilde{d}_i > d_i$, and $C(d_i)$ must be less than zero if $\tilde{d}_i < d_i$. Figure 7 illustrates this, along with an eigenvalue which remains unchanged at $\lambda = d_3$.

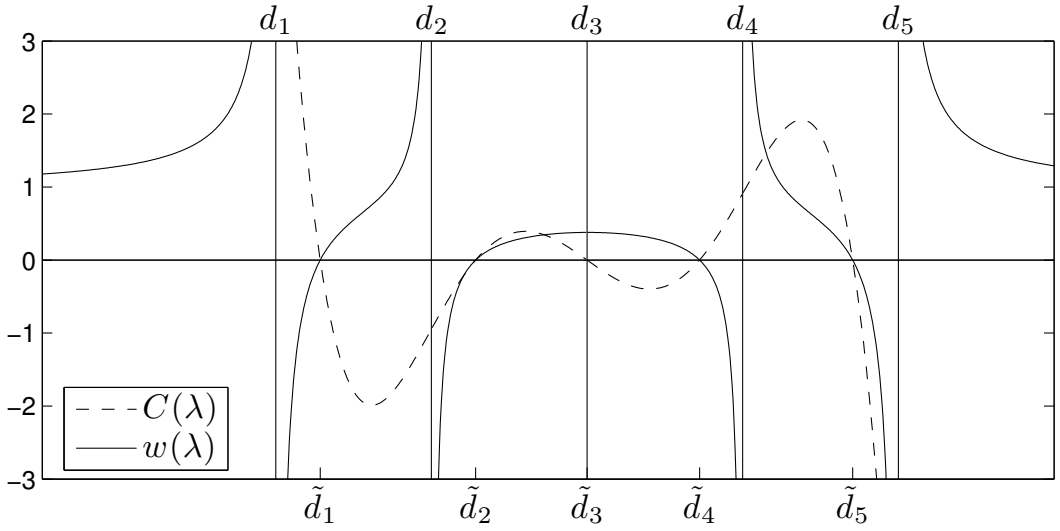


Figure 7. An example to illustrate the rank-two secular function and characteristic polynomial of $\tilde{G} = D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H$, where the eigenvalue at d_3 does not change, but the corresponding eigenvector changes. The values used to create this figure were $D = \text{diag}(1, 2, 3, 4, 5)$, $\boldsymbol{a} = [0.13, 0.26, 0.4, 0.53, 0.67]^T$, and $\boldsymbol{b} = [0.66, 0.53, b_3, 0.26, 0.19]^T$, where $b_3 = 0.39137187095400$. The eigenvalues of $\tilde{G}$ are $\tilde{D} \simeq \text{diag}(1.29, 2.28, 3.0, 3.72, 4.71)$. Due to the pole and zero of $w(\lambda)$ at $\lambda = d_3$, the characteristic polynomial must be used to determine that the eigenvalue did not change.

It can be seen from (117) that $C(-\infty) = +\infty$, which means that the slope of $C(\lambda)$ at $\tilde{d}_1$ (the smallest root) must be negative. Since we know that we have exactly n real roots, and that the sign of the slope of $C(\lambda)$ must alternate at each root, the sign of the slope of $C(\tilde{d}_i)$ will be

$$s_r(i) = (-1)^i. \tag{131}$$

When we multiply $s_r(i)$ by $s_e(i)$ and negate the result, we get

$$s_p(i) = \text{sgn}\left(|b_i|^2 - |a_i|^2 - \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{|a_i b_j - a_j b_i|^2}{(d_j - d_i)} \right), \tag{132}$$

which will evaluate to 1 when $\tilde{d}_i$ is greater than d_i, 0 when $\tilde{d}_i$ equals d_i, and -1

when $\tilde{d}_i$ is less than d_i. To take into account duplicate eigenvalues, we define

$$s_t(i) = \begin{cases} -1, & \text{if } d_i = d_{i+1} \\ 1, & \text{if } d_i = d_{i-1} \\ s_p(i), & \text{if } d_{i-1} \neq d_i \neq d_{i+1} \end{cases}, \tag{133}$$

where the terms for the duplicate eigenvalues are based on (126). If we define the binary value $s_b(i) = (s_t(i) + 1)/2$, which will evaluate to 1 when $\tilde{d}_i > d_i$, and 0 when $\tilde{d}_i < d_i$, we can shrink the intervals for eigenvalues that have changed to be

$$d_{i-1+s_b(i)} < \tilde{d}_i < d_{i+s_b(i)}, \qquad i = 1 \cdots n, \tag{134}$$

which is a bound between two adjacent old eigenvalues.

Splitting the shared intervals

Now that we have separated the roots into intervals of adjacent old eigenvalues, a few of these intervals may have two roots. This will happen whenever the ith root is greater than d_i and the $(i+1)$th root is less than d_{i+1}. An easy way to test for this condition is to check if $s_b(i) - s_b(i+1)$ is equal to 1. When we find an interval that contains two roots, we need to determine if the root is a multiple root or two separate roots. If it is two separate roots, we need to separate the interval into two disjoint intervals containing one root each by finding any point between the two roots. Figure 6 shows two roots in the interval between d_2 and d_3.

From (120), we know that the rank-two secular equation is just the product of two rank-one secular equations of opposite slope, plus a positive term. The product of the two rank-one secular equations will be positive in the interval between their roots because they will have the same sign in that interval, and negative outside of that interval because their signs will differ. Since the term $|w_{ab}(\lambda)|^2$ from (120) is positive for all λ, the rank-two secular equation will be positive between its two roots, and negative outside. If we can find any λ between d_i and d_{i+1} that makes $w(\lambda)$ evaluate to a non-negative value, this point will separate the two roots.

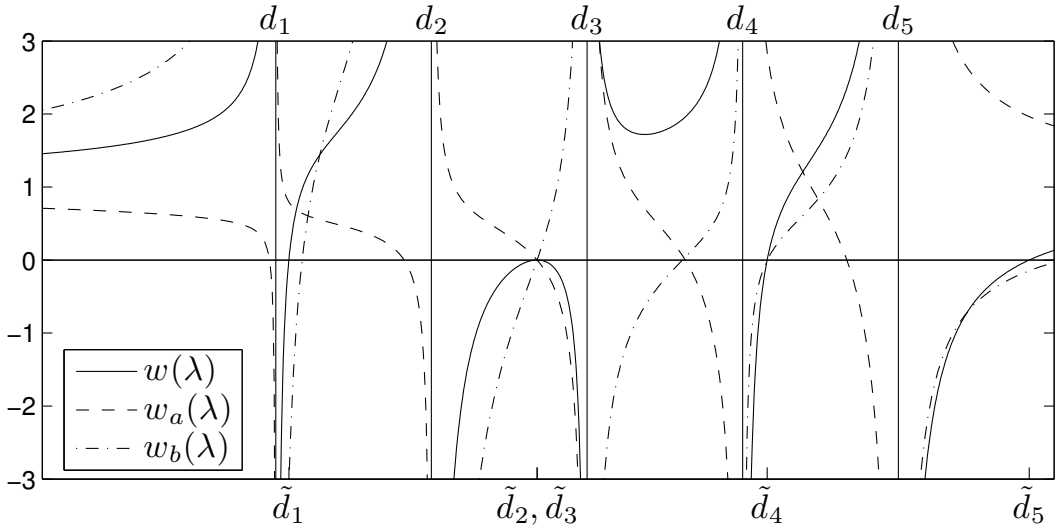


Figure 8. The secular functions of $\tilde{G} = D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H$, $\tilde{G}_a = D - \boldsymbol{a}\boldsymbol{a}^H$, and $\tilde{G}_b = D + \boldsymbol{b}\boldsymbol{b}^H$, which illustrate the creation of a duplicate eigenvalue at $\lambda = \tilde{d}_2 = \tilde{d}_3 \simeq 2.68$. The values used to generate this figure were $D = \mathrm{diag}(1, 2, 3, 4, 5)$, $\boldsymbol{a} = [0.15, 0.3, 0.45, 0.6, 0.75]^T$, and $\boldsymbol{b} = U_a \ddot{\boldsymbol{b}}$, where U_a are the eigenvectors of $D - \boldsymbol{a}\boldsymbol{a}^H$ in ascending order of eigenvalues, and $\ddot{\boldsymbol{b}} = [0.625, \ddot{b}_2, 0, 0.75, 0.625]^T$, where $\ddot{b}_2 = 1.15485665309790$. The eigenvalues of $\tilde{G}$ are $\tilde{D} \simeq \mathrm{diag}(1.08, 2.68, 2.68, 4.16, 5.84)$. Note that the roots of $w_a(\lambda)$, which are the eigenvalues of $\tilde{G}_a$, must be less than the old eigenvalues, while the roots of $w_b(\lambda)$, which are the eigenvalues of $\tilde{G}_b$, must be greater than the old eigenvalues.

The only way to have a multiple root at $\tilde{d}_i$ and $\tilde{d}_{i+1}$, is if the ith root of the secular equation for $D + \boldsymbol{b}\boldsymbol{b}^H$, which is $w_b(\lambda)$, is the same as the $(i+1)$th root of the secular equation for $D - \boldsymbol{a}\boldsymbol{a}^H$, which is $w_a(\lambda)$, because their product will have no effect on the location of their root in (120). Because the term $|w_{ab}(\lambda)|^2$ from (120) is non-negative for all λ, and the slope of $w(\lambda)$ is non-negative at the smaller of the two roots and negative at the larger of the two roots, the only effect that adding $|w_{ab}(\lambda)|^2$ can have is to separate the two roots further, therefore $w_{ab}(\lambda)$ must equal zero at the multiple root. Figure 8 shows the creation of a multiple root at $\lambda = \tilde{d}_2 = \tilde{d}_3$.

Because of the effect of adding $|w_{ab}(\lambda)|^2$, both the ith root of $w_b(\lambda)$ and the $(i+1)$th root of $w_a(\lambda)$ must be between the two roots of $w(\lambda)$ when they are not

identical. Since we only need to find any point between the two roots of $w(\lambda)$ to separate them, we can converge towards either of these roots, and stop when $w(\lambda)$ evaluates to a non-negative value. A good method to converge towards these roots is given in [3]. If the roots are not too close, we should find a non-negative value with few, if any iterations. If the roots are identical, we will have converged on the actual duplicate root of $w(\lambda)$.

2.5.4 Calculating the roots of $w(\lambda)$

Now that we have unique bounds for each root of $w(\lambda)$, we need to find those roots. This can be done using Newton's method with protection by bisection.

There are three cases where bisection will be needed. First, when there is only one root in an interval between two adjacent old eigenvalues, it is possible that there is a local minima, which can cause Newton's method to move in the wrong direction. Second, when there are two roots in an interval between two adjacent old eigenvalues, one of the new intervals may contain both positive and negative slopes. Third, when Newton's method produces a new estimate outside the interval.

To use bisection effectively, we need to know the sign of the slope of $w(\lambda)$ at the root, so we can shrink the bounds as we converge on the root. The sign of the slope of $w(\lambda)$ at its roots is the same as the sign of the slope of $C(\lambda)$ at its roots, except the sign is inverted at every old eigenvalue. We already have term for this, and it is $s_t(i)$ from (133).

Since the roots of $w(\lambda)$ are just the squares of the singular values of $\tilde{M}$, we now effectively have $\tilde{\Sigma}_M$.

2.6 The eigenvectors of $\tilde{G}$

Now that we have the eigenvalues of $\tilde{G}$, we will now determine the corresponding eigenvectors. The eigenvectors which correspond to eigenvalues that have not changed and eigenvectors which correspond to duplicate eigenvalues must be calculated differently from the other eigenvectors.

2.6.1 Eigenvectors of changed, non-duplicate eigenvalues

The method presented in this section for calculating eigenvectors for the rank two case uses the basic idea from [3] for calculating the eigenvectors for the rank-one case.

If $\boldsymbol{x}_i$ is the ith right eigenvector of $\tilde{G} = D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H$, multiplied by a real or complex scalar value, and $\tilde{d}_i$ is the corresponding ith eigenvalue, where $\tilde{d}_i \neq d_i$, then from the definition of a right eigenvector [6], we can say

$$\left(D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H\right)\boldsymbol{x}_i = \tilde{d}_i\boldsymbol{x}_i. \tag{135}$$

Rearranging some terms, and substituting $T_{\tilde{d}_i} = (D - \tilde{d}_i I)$ from (112), we get

$$T_{\tilde{d}_i}\boldsymbol{x}_i = \left(\boldsymbol{a}\boldsymbol{a}^H - \boldsymbol{b}\boldsymbol{b}^H\right)\boldsymbol{x}_i. \tag{136}$$

We know that $T_{\tilde{d}_i}$ is invertible, because in this section we are only calculating eigenvectors that correspond to changed eigenvalues. Multiplying both sides of (136) by the inverse of $T_{\tilde{d}_i}$, we get

$$\boldsymbol{x}_i = T_{\tilde{d}_i}^{-1}\left(\boldsymbol{a}\boldsymbol{a}^H - \boldsymbol{b}\boldsymbol{b}^H\right)\boldsymbol{x}_i. \tag{137}$$

If we define the two scalar values $c_1 = \boldsymbol{a}^H\boldsymbol{x}_i$ and $c_2 = -\boldsymbol{b}^H\boldsymbol{x}_i$, we can write (137) as a linear combination of $\boldsymbol{a}$ and $\boldsymbol{b}$

$$\boldsymbol{x}_i = T_{\tilde{d}_i}^{-1}\left(c_1\boldsymbol{a} + c_2\boldsymbol{b}\right). \tag{138}$$

If we multiply both sides of (138) by $\boldsymbol{a}^H$ from the left, we get

$$\boldsymbol{a}^H\boldsymbol{x}_i = \boldsymbol{a}^H T_{\tilde{d}_i}^{-1}\left(c_1\boldsymbol{a} + c_2\boldsymbol{b}\right), \tag{139}$$

where $\boldsymbol{a}^H\boldsymbol{x}_i$ on the left side is c_1. Solving for c_2 we get

$$c_2 = c_1 \frac{1 - \boldsymbol{a}^H T_{\tilde{d}_i}^{-1}\boldsymbol{a}}{\boldsymbol{a}^H T_{\tilde{d}_i}^{-1}\boldsymbol{b}} = c_1 \frac{w_a(\tilde{d}_i)}{w_{ab}(\tilde{d}_i)}, \tag{140}$$

where $w_a(\tilde{d}_i)$ and $w_{ab}(\tilde{d}_i)$ are from (121). Plugging (140) into (138) we get

$$\frac{\boldsymbol{x}_i}{c_1} = T_{\tilde{d}_i}^{-1}\left(\boldsymbol{a} + \frac{1 - \boldsymbol{a}^H T_{\tilde{d}_i}^{-1}\boldsymbol{a}}{\boldsymbol{a}^H T_{\tilde{d}_i}^{-1}\boldsymbol{b}}\,\boldsymbol{b}\right) = T_{\tilde{d}_i}^{-1}\left(\boldsymbol{a} + \frac{w_a(\tilde{d}_i)}{w_{ab}(\tilde{d}_i)}\,\boldsymbol{b}\right). \tag{141}$$

Now at this point we have $\boldsymbol{x}_i/c_1$, which is a vector that points in the same direction as the ith new eigenvector, but is not unit length, therefore if we divide it by its norm, we have our eigenvector,

$$\tilde{\boldsymbol{u}}_i = \frac{\boldsymbol{x}_i/c_1}{\|\boldsymbol{x}_i/c_1\|} = \frac{T_{\tilde{d}_i}^{-1}\left(\boldsymbol{a} + \frac{w_a(\tilde{d}_i)}{w_{ab}(\tilde{d}_i)}\,\boldsymbol{b}\right)}{\left\|T_{\tilde{d}_i}^{-1}\left(\boldsymbol{a} + \frac{w_a(\tilde{d}_i)}{w_{ab}(\tilde{d}_i)}\,\boldsymbol{b}\right)\right\|}. \tag{142}$$

Note that we could have solved for c_2 instead of c_1 by multiplying both sides of (138) by $\boldsymbol{b}^H$ instead of $\boldsymbol{a}^H$.

2.6.2 Eigenvectors of unchanged eigenvalues

When $\tilde{d}_i$ equals d_i, (remember both a_i and b_i are not zero because we already removed that case when we deflated), the matrix $T_{\tilde{d}_i}$ from (142) is not invertible because it will have a zero on the diagonal. Therefore, we must use an alternative method to calculate the eigenvector. We will start from (136), where $\boldsymbol{x}_i$ is again the ith right eigenvector multiplied by a real or complex scalar value. If we look at the ith element from each side of (136), which corresponds to the unchanged eigenvalue, we get

$$\left(d_i - \tilde{d}_i\right) x_{i,i} = \left(a_i\boldsymbol{a}^H - b_i\boldsymbol{b}^H\right)\boldsymbol{x}_i, \tag{143}$$

where $x_{j,i}$ is the jth element of $\boldsymbol{x}_i$. Since the ith eigenvalue is unchanged, $d_i = \tilde{d}_i$; therefore, the left side is zero, and we can write

$$\boldsymbol{b}^H\boldsymbol{x}_i = \frac{a_i}{b_i}\boldsymbol{a}^H\boldsymbol{x}_i. \tag{144}$$

Combining (136) and (144) we get

$$T_{\tilde{d}_i}\boldsymbol{x}_i = \left(\boldsymbol{a} - \frac{a_i}{b_i}\boldsymbol{b}\right)\boldsymbol{a}^H\boldsymbol{x}_i, \tag{145}$$

If we remove the ith element of $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{x}_i$, and the ith row and column of $T_{\tilde{d}_i}$, and call these $\dot{\boldsymbol{a}}$, $\dot{\boldsymbol{b}}$, $\dot{\boldsymbol{x}}_i$, and $\dot{T}_{\tilde{d}_i}$ respectively, (145) can be rewritten as

$$\dot{T}_{\tilde{d}_i}\dot{\boldsymbol{x}}_i = \left(\dot{\boldsymbol{a}} - \frac{a_i}{b_i}\dot{\boldsymbol{b}}\right)\boldsymbol{a}^H\boldsymbol{x}_i. \tag{146}$$

Because we removed the zero from the diagonal of $T_{\tilde{d}_i}$ when we created $\dot{T}_{\tilde{d}_i}$, we can now invert $\dot{T}_{\tilde{d}_i}$. Multiplying (146) by the inverse of $\dot{T}_{\tilde{d}_i}$, and defining the scalar value $c_1 = \boldsymbol{a}^H\boldsymbol{x}_i$, we get

$$\frac{\dot{\boldsymbol{x}}_i}{c_1} = \dot{T}_{\tilde{d}_i}^{-1}\left(\dot{\boldsymbol{a}} - \frac{a_i}{b_i}\dot{\boldsymbol{b}}\right), \tag{147}$$

which gives us all of the elements of $\boldsymbol{x}_i/c_1$ except the ith element.

Next, we need to calculate $x_{i,i}/c_1$. Rewriting (144) as $b_i\boldsymbol{b}^H\boldsymbol{x}_i = a_i\boldsymbol{a}^H\boldsymbol{x}_i$, then expanding the vector product as a summation, we get

$$b_i\sum_{j=1}^{n} b_j^* x_{j,i} = a_i\sum_{j=1}^{n} a_j^* x_{j,i}. \tag{148}$$

Dividing both sides of (148) by c_1 and moving the ith element out of both summations, we get

$$b_i b_i^*\frac{x_{i,i}}{c_1} + b_i\sum_{\substack{j=1\\j\neq i}}^{n} b_j^*\frac{x_{j,i}}{c_1} = a_i a_i^*\frac{x_{i,i}}{c_1} + a_i\sum_{\substack{j=1\\j\neq i}}^{n} a_j^*\frac{x_{j,i}}{c_1}. \tag{149}$$

Solving for $x_{i,i}/c_1$, we get

$$\frac{x_{i,i}}{c_1} = \frac{1}{a_i a_i^* - b_i b_i^*}\left(b_i\sum_{\substack{j=1\\j\neq i}}^{n} b_j^*\frac{x_{j,i}}{c_1} - a_i\sum_{\substack{j=1\\j\neq i}}^{n} a_j^*\frac{x_{j,i}}{c_1}\right), \tag{150}$$

which can be written as

$$\frac{x_{i,i}}{c_1} = \frac{b_i\dot{\boldsymbol{b}}^H\dot{\boldsymbol{x}}_i/c_1 - a_i\dot{\boldsymbol{a}}^H\dot{\boldsymbol{x}}_i/c_1}{|a_i|^2 - |b_i|^2} = \left(\frac{b_i\dot{\boldsymbol{b}}^H - a_i\dot{\boldsymbol{a}}^H}{|a_i|^2 - |b_i|^2}\right)\frac{\dot{\boldsymbol{x}}_i}{c_1}. \tag{151}$$

Inserting $x_{i,i}/c_1$ from (151) into the ith position of $\dot{\boldsymbol{x}}_i/c_i$ from (147), we get $\boldsymbol{x}_i/c_1$, which is a vector that points in the same direction as the ith new eigenvector, but is not unit length. If we divide it by its norm, we have our eigenvector,

$$\tilde{\boldsymbol{u}}_i = \frac{\boldsymbol{x}_i/c_1}{\|\boldsymbol{x}_i/c_1\|}. \tag{152}$$

2.6.3 Eigenvectors of duplicate eigenvalues

When the ith new eigenvalue, $\tilde{d}_i$, is the same as the $(i+1)$th new eigenvalue, $\tilde{d}_{i+1}$, section 2.5.3 shows us that the terms $w_a(\lambda)$, $w_b(\lambda)$, and $w_{ab}(\lambda)$ from (119) must all be zero at these eigenvalues. This prevents us from using (142) because of the $w_a(\tilde{d}_i)/w_{ab}(\tilde{d}_i)$ term.

Eq. (127) shows that the only way to create a duplicate eigenvalue in a rank-one perturbation is to have an element of $\boldsymbol{a}$ or $\boldsymbol{b}$ that is zero. This means that to create a duplicate eigenvalue in the deflated rank-two case, there must be a zero in the *second* (rotated) rank one modification. Figure 8 illustrates what is happening when a duplicate eigenvalue is created for $\tilde{d}_2 = \tilde{d}_3$ in a rank-two modification.

Because of the zero in the $(i+1)$th element of $\ddot{\boldsymbol{b}} = U_a^H \boldsymbol{b}$, the $(i+1)$th eigenvalue and eigenvector of $D - \boldsymbol{a}\boldsymbol{a}^H$ will not change during the second rank one modification. There will also be a zero in the ith element of $\ddot{\boldsymbol{a}} = U_b^H \boldsymbol{a}$, which will cause the ith eigenvalue and eigenvector of $D + \boldsymbol{b}\boldsymbol{b}^H$ to remain unchanged during the second rank-one modification when the rank-one modifications are done in the reverse order. This means that we can use the rank-one formula from [3] to determine two non-orthogonal, non-normal eigenvectors, $\boldsymbol{x}_i = T_{\tilde{d}_i}^{-1}\boldsymbol{a}$ and $\boldsymbol{x}_{i+1} = T_{\tilde{d}_i}^{-1}\boldsymbol{b}$. In section 2.4.2 we showed that when there are duplicate eigenvalues in a matrix, the corresponding eigenvectors are contained in a subspace of arbitrary rotation. Therefore, if we define our eigenvectors as

$$\tilde{\boldsymbol{u}}_i = \frac{T_{\tilde{d}_i}^{-1}\boldsymbol{a}}{\|T_{\tilde{d}_i}^{-1}\boldsymbol{a}\|}, \qquad \tilde{\boldsymbol{u}}_{i+1} = \frac{\left(I - \tilde{\boldsymbol{u}}_i\tilde{\boldsymbol{u}}_i^H\right) T_{\tilde{d}_i}^{-1}\boldsymbol{b}}{\left\|\left(I - \tilde{\boldsymbol{u}}_i\tilde{\boldsymbol{u}}_i^H\right) T_{\tilde{d}_i}^{-1}\boldsymbol{b}\right\|}, \tag{153}$$

they will be orthogonal and normalized.

2.6.4 The singular vectors of $\tilde{M}$

After calculating $\boldsymbol{u}_i$ for $i = 1 \cdots n$, we have the matrix of eigenvectors of $\tilde{G}$,

$$\tilde{U} = \left[\begin{array}{ccccccc} \tilde{\boldsymbol{u}}_1 & \tilde{\boldsymbol{u}}_2 & \tilde{\boldsymbol{u}}_3 & \cdots & \tilde{\boldsymbol{u}}_{n-1} & \tilde{\boldsymbol{u}}_n \end{array} \right]. \tag{154}$$

Using (103), we can get the left singular vectors of $\tilde{M}$ as $\tilde{U}_M = U_M \tilde{U}$.

2.7 Putting it all together

In this section, we list the specific steps required to calculate the eigendecomposition of the matrix $\tilde{G} = D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H$ using the method presented in this manuscript. In Table 6 we show how to efficiently sort and deflate $\tilde{G}$ as described in section 2.4, which is necessary before beginning the eigendecomposition. In Table 7, we list the specific steps required to calculate the eigendecomposition of the sorted and deflated version of the matrix $\tilde{G}$. These steps can be thought of as summary of the results presented in this manuscript minus the theory.

When using this method to update the SVD of a matrix in which one column has been replaced, we start with the singular values, Σ_M, and left singular vectors, U_M, of the original matrix, along with the column we are discarding, $\boldsymbol{m}_1$, and the column we are adding $\boldsymbol{m}_{c+1}$. We set

$$D = \Sigma_M^2, \qquad \boldsymbol{a} = U_M^H \boldsymbol{m}_1, \qquad \boldsymbol{b} = U_M^H \boldsymbol{m}_{c+1},$$

then calculate the eigendecomposition of $\tilde{G} = D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H = \tilde{U}\tilde{D}\tilde{U}^H$ using the steps in Tables 6 and 7. Our new singular values and left singular vectors are then

$$\tilde{\Sigma}_M = \sqrt{\tilde{D}}, \qquad \tilde{U}_M = U_M \tilde{U},$$

respectively. The new right singular vectors can be calculated as $\tilde{V}_M = \tilde{M}^H \tilde{U}_M \tilde{\Sigma}_M^{-1}$ if needed.

Table 6. The steps to efficiently sort and deflate $\tilde{G} = D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H$ prior to determining its eigendecomposition. $\tilde{G}$ consists of the diagonal matrix $D \in \mathbb{R}^{n\times n}$, and the vectors $\boldsymbol{a} \in \mathbb{C}^n$ and $\boldsymbol{b} \in \mathbb{C}^n$.

1)	Create D' by sorting the elements on the diagonal of D in ascending order. Create a length n vector $\boldsymbol{s}$, that contains the indices of the sorted elements of D, such that $d'_i = d_{s_i}$. Create $\boldsymbol{a}'$ and $\boldsymbol{b}'$ by setting $a'_i = a_{s_i}$ and $b'_i = b_{s_i}$ for each i.
2)	Create an $n \times n$ matrix I_r, and initialize it to the identity matrix. Create the vectors $\dot{\boldsymbol{a}}$ and $\dot{\boldsymbol{b}}$, and initialize them to $\boldsymbol{a}'$ and $\boldsymbol{b}'$. For each set of duplicate eigenvalues, d'_i through d'_{i+k-1}, take the QR factorization of $$QR = \begin{bmatrix} a'_i & a'_{i+1} & a'_{i+2} & \cdots & a'_{i+k-1} \\ b'_i & b'_{i+1} & b'_{i+2} & \cdots & b'_{i+k-1} \end{bmatrix}^T.$$ Insert the first column of R into $\dot{a}_i$ through $\dot{a}_{i+k-1}$, the second column of R into $\dot{b}_i$ through $\dot{b}_{i+k-1}$, and Q into I_r at the i, i through $i+k-1, i+k-1$ block.
3)	Create the length n vector $\boldsymbol{z}$, and initialize it to $z_i = i$. Find each i, where $\dot{a}_i = \dot{b}_i = 0$, then for each of these i, remove the ith element of $\boldsymbol{z}$, and append it to the vector $\tilde{\boldsymbol{z}}$. If we found n_d values of i where $\dot{a}_i = \dot{b}_i = 0$, $\boldsymbol{z}$ will now be length $n - n_d$, and $\tilde{\boldsymbol{z}}$ will now be length n_d.
4)	Create the $n \times n_d$ matrix $\tilde{U}_d$ by selecting the columns of I_r which correspond to the values in $\tilde{\boldsymbol{z}}$. Create the $n_d \times n_d$ diagonal matrix $\tilde{D}_d$ by selecting the elements on the diagonal of D' which correspond to the values in $\tilde{\boldsymbol{z}}$.
5)	Create the $n \times n - n_d$ matrix I_d, by selecting the columns of I_r which correspond to the values in $\boldsymbol{z}$.
6)	Create the $n - n_d \times n - n_d$ diagonal matrix D'', by selecting the elements on the diagonal of D' which correspond to the values in $\boldsymbol{z}$. Create the length $n - n_d$ vectors $\boldsymbol{a}''$ and $\boldsymbol{b}''$ by selecting the elements of $\dot{\boldsymbol{a}}$ and $\dot{\boldsymbol{b}}$ which correspond to the values in $\boldsymbol{z}$.
7)	Calculate the eigendecomposition, $\tilde{U}''\tilde{D}''\tilde{U}''^H = D'' - \boldsymbol{a}''\boldsymbol{a}''^H + \boldsymbol{b}''\boldsymbol{b}''^H$, using the steps in table 7.
8)	Create the matrix $\tilde{U}' = [\tilde{U}_d \mid I_d\tilde{U}'']$. Note that each row of $I_d\tilde{U}''$ is either all zeros, a row directly transfered from $\tilde{U}''$, or in the case of duplicate old eigenvalues, a linear combination of up to two rows of $\tilde{U}''$.
9)	Define a vector $\tilde{\boldsymbol{s}}$, whose elements are $\tilde{s}_{s_i} = i$. Undo the sorting from step 1 by setting the ith row of $\tilde{U}$ to the $\tilde{s}_i$th row of $\tilde{U}'$, and we have our eigenvectors.
10)	Create the matrix $\tilde{D}$ by concatenating the diagonal matrices $\tilde{D}_d$ and $\tilde{D}''$ as in (110), and we have our eigenvalues.

Table 7. The steps to calculate the eigendecomposition of the sorted and deflated $\tilde{G} = D - \boldsymbol{a}\boldsymbol{a}^H + \boldsymbol{b}\boldsymbol{b}^H$.

1)	Calculate $s_t(i)$ for $i = 1 \cdots n$ using (132) and (133). For any i, where $s_t(i) = 0$, set $\tilde{d}_i = d_i$, because that eigenvalue doesn't change.
2)	Calculate $s_b(i) = (s_t(i) + 1)/2$, for $i = 1 \cdots n$.
3)	Set the bounds for each new root to $(d_{i-1+s_b(i)} < \tilde{d}_i < d_{i+s_b(i)})$ from (134), where $d_0 = d_1 - \|\boldsymbol{a}\|_2^2$ and $d_{n+1} = d_n + \|\boldsymbol{b}\|_2^2$.
4)	Find each i where $s_b(i) - s_b(i+1)$ is equal to 1. These i will have two roots between d_i and d_{i+1} which we will need to separate. For these values of i, find a value of λ between d_i and d_{i+1} where $w(\lambda)$ is positive. Start with the midpoint, $\lambda = (d_i + d_{i+1})/2$, and use Newton's method protected by bisection on $w_a(\lambda)$ from (119) to determine new values for λ. If we converge to the root of $w_a(\lambda)$ without $w(\lambda)$ evaluating to a positive value, then we have a duplicate root at $\tilde{d}_i = \tilde{d}_{i+1} = \lambda$. At any time that $w(\lambda)$ evaluates to a positive value, change the upper bound of the ith root and the lower bound of the $(i+1)$th root to λ, because that λ separates the two roots.
5)	For each of the eigenvalues of $\tilde{G}$ that we haven't found yet (we found the unchanged ones in step 2, and the duplicate ones in step 5), calculate the bounded root of $w(\lambda)$ using Newton's method with protection by bisection as described in §2.5.4.
6)	For each of the eigenvalues that changed, and are not duplicate, calculate the corresponding eigenvector using (142).
7)	For each pair of duplicate eigenvalues, calculate the corresponding two eigenvectors using (153).
8)	For each eigenvalue that didn't change, calculate the corresponding eigenvector using (147), (151), and (152).

2.8 Concluding Remarks

The method presented in this manuscript is intended to show how the eigendecomposition of a rank-two modification to a diagonal matrix can be computed directly in $O(n^2)$ time. As an algorithm, there are numerical stability issues for closely spaced eigenvalues, similar to those in the rank-one case, which need to be addressed before this algorithm can be used as a robust method for calculating the eigendecomposition in all cases. We have not spent much time looking into numerical roundoff issues related to this algorithm, as the basic theory as presented in this manuscript is more understandable in a theoretical sense. It is possibly more

efficient and/or accurate to use the rank-one method from [3] twice, then calculate the eigenvectors using the method described in section 2.6. The method from [4, 5] might be able to be adapted to the calculation of the eigenvectors from section 2.6.

As an analysis tool, the rank-two secular function gives insight into what happens to the eigenspace of a matrix when a column is replaced, and this is the reason that the authors developed this method in the first place. The relation of the rank-two secular function to the rank-one secular function is interesting, although not surprising.

2.9 Evaluating the determinant in $C(\lambda)$

In this section, we give the details for writing $C(\lambda)$ from (115) in the simple closed form of (116). We will start by using the fact that the determinant of an $n \times n$ matrix can be written as the sum of n, $(n-1) \times (n-1)$ determinants,

$$|A| = \sum_{j=1}^{n} (-1)^{j+1} a_{1,j} \left| A_{1,j} \right|, \tag{155}$$

where $A_{1,j}$ is an $(n-1) \times (n-1)$ matrix obtained by deleting the first row and jth column of A [6]. The properties of determinants that will be used here are

- the determinant of a matrix which has a row or column consisting of all zeros is zero,
- the determinant of a matrix which can be written as either an upper or lower triangular matrix is the product of the diagonal terms,
- swapping any two rows or columns in the matrix changes the sign of the determinant, and
- the transpose of a matrix has the same determinant as the original matrix.

Using (155), we can write the determinant of an $n \times n$ matrix with nonzero elements only on the diagonal, the first row, and the first column, as the scaled sum of the

determinants of n, $(n-1)\times(n-1)$ triangular matrices. The determinant of a matrix of this form can be written as

$$A_1 = \begin{vmatrix} w_1 & z_2 & z_3 & \cdots & z_n \\ y_2 & x_2 & 0 & \cdots & 0 \\ y_3 & 0 & x_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_n & 0 & 0 & \cdots & x_n \end{vmatrix} = w_1 \prod_{i=2}^{n} x_i - \sum_{j=2}^{n} y_j z_j \left(\prod_{\substack{i=2 \\ i\neq j}}^{n} x_i \right). \tag{156}$$

The reason there is no alternating sign in the sum is due to row swapping to make the sub-matrices triangular. Also using (155) we can show that a matrix with nonzero elements only on the diagonal, the first row, and the first *two* columns can be written as the scaled sum of determinants of one matrix of the form of (156) plus $n-1$ triangular matrices. The determinant of a matrix of this form can be written as

$$A_2 = \begin{vmatrix} w_{1,1} & w_{1,2} & z_3 & z_4 & \cdots & z_n \\ w_{2,1} & w_{2,2} & 0 & 0 & \cdots & 0 \\ y_3 & u_3 & x_3 & 0 & \cdots & 0 \\ y_4 & u_4 & 0 & x_4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ y_n & u_n & 0 & 0 & \cdots & x_n \end{vmatrix}, \tag{157}$$

$$\begin{aligned} A_2 \;=\; & w_{1,1}w_{2,2}\left(\prod_{i=3}^{n} x_i\right) - w_{1,2}w_{2,1}\left(\prod_{i=3}^{n} x_i\right) \\ & + w_{2,1}\sum_{j=3}^{n} u_j z_j \left(\prod_{\substack{i=3 \\ i\neq j}}^{n} x_i\right) - w_{2,2}\sum_{j=3}^{n} y_j z_j \left(\prod_{\substack{i=3 \\ i\neq j}}^{n} x_i\right). \end{aligned} \tag{158}$$

Finally, when we have a matrix with nonzero elements only on the diagonal, the first *two* rows, and the first *two* columns, the determinant can be written as the scaled sum of determinants of two matrices of the form of (156) plus $n-2$ matrices

of the form of (157). The determinant of a matrix of this form can be written as

$$A_3 = \begin{vmatrix} w_{1,1} & w_{1,2} & z_3 & z_4 & \cdots & z_n \\ w_{2,1} & w_{2,2} & v_3 & v_4 & \cdots & v_n \\ y_3 & u_3 & x_3 & 0 & \cdots & 0 \\ y_4 & u_4 & 0 & x_4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ y_n & u_n & 0 & 0 & \cdots & x_n \end{vmatrix}, \tag{159}$$

$$\begin{aligned} A_3 = \quad & \left(\prod_{i=3}^{n} x_i\right)\left(w_{1,1}w_{2,2} - w_{1,2}w_{2,1} + \sum_{j=3}^{n} \frac{v_j}{x_j}\left(y_j w_{1,2} - w_{1,1}u_j\right)\right. \\ & \left. + \sum_{j=3}^{n} \frac{z_j}{x_j}\left(w_{2,1}u_j - y_j w_{2,2}\right) + \sum_{j=3}^{n} \frac{v_j}{x_j} \sum_{l=3}^{n} \frac{z_l}{x_l}\left(y_l u_j - y_j u_l\right)\right). \end{aligned} \tag{160}$$

Since (115) has the same form as (159), we can just plug the values into (160). The $w_{i,j}$ terms in (160) are not special, but just an artifact from converting (113) to (115), so after some algebraic manipulation we get (116).

List of References

[1] J. H. Wilkinson, *The Algebraic Eigenvalue Problem.* London, UK: Oxford Univ. Press, 1965.

[2] G. H. Golub, "Some modified matrix eigenvalue problems," *SIAM Review*, vol. 15, no. 2, pp. 318–334, Apr. 1973.

[3] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, "Rank-one modification of the symmetric eigenproblem," *Numerische Mathematik*, vol. 31, pp. 31–48, 1978.

[4] M. Gu and S. C. Eisenstat, "A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem," *SIAM Journal on Matrix Analysis and Applications*, vol. 15, no. 4, pp. 1266–1276, Oct. 1994.

[5] M. Gu and S. C. Eisenstat, "Downdating the singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 16, no. 3, pp. 793–810, July 1995.

[6] G. H. Golub and C. F. van Loan, *Matrix Computations, 3rd ed.* Baltimore, MD: Johns Hopkins Univ. Press, 1996.

MANUSCRIPT 3

A Subspace Tracking Algorithm Using the Fast Fourier Transform

3.1 Abstract

At ICASSP '97 and in the July 1999 IEEE Transactions on Signal Processing Real, Tufts, and Cooley presented an algorithm for fast tracking of a signal subspace or interference subspace for application in adaptive detection or estimation. For cases in which the signal matrix is formed from a single-channel discrete-time signal, we show how one can further reduce computation in the FAST algorithm by using the *fast Fourier transform* (FFT).

3.2 The Subspace Tracking Algorithm

At each time step, the FAST algorithm [1] computes an approximate *singular value decomposition* (SVD) of an $r \times c$ signal matrix,

$$\boldsymbol{M}_{new} = \left[\begin{array}{cccc} \boldsymbol{m}_2 & \boldsymbol{m}_3 & \cdots & \boldsymbol{m}_{c+1} \end{array} \right], \tag{161}$$

where each $\boldsymbol{m}_j$ is an r-element column vector. The approximation is based on the use of a smaller dimensional subspace that contains most of the signal. This subspace comes from the previous iteration, which computed the approximate SVD of the matrix

$$\boldsymbol{M}_{old} = \left[\begin{array}{cccc} \boldsymbol{m}_1 & \boldsymbol{m}_2 & \cdots & \boldsymbol{m}_c \end{array} \right], \tag{162}$$

which shares $c-1$ of its c columns with $\boldsymbol{M}_{new}$.

This note shows how the FFT algorithm may be used to reduce computation in the FAST algorithm for the situation in which $\boldsymbol{M}_{old}$ and $\boldsymbol{M}_{new}$ are formed from a single-channel discrete-time signal [2, 3, 4]. In this case $\boldsymbol{M}_{old}$ and $\boldsymbol{M}_{new}$ are $r \times c$

Hankel-like (Hankel if $r = c$) matrices of the form,

$$\boldsymbol{M}_{new} = \begin{bmatrix} d_2 & d_3 & \cdots & d_{c+1} \\ d_3 & d_4 & \cdots & d_{c+2} \\ \vdots & \vdots & & \vdots \\ d_{r+1} & d_{r+2} & \cdots & d_{r+c} \end{bmatrix}. \tag{163}$$

The $N = r + c - 1$ unique elements of $\boldsymbol{M}_{new}$ can be written as the $N \times 1$ column vector

$$\boldsymbol{d}_{new} = \begin{bmatrix} d_2 & d_3 & \cdots & d_{N+1} \end{bmatrix}^T. \tag{164}$$

At the beginning of the current iteration, one has k principal left singular vectors of $\boldsymbol{M}_{old}$ in the columns of the matrix

$$\boldsymbol{U}_{old} = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_k \end{bmatrix}, \tag{165}$$

as well as the matrix $\boldsymbol{M}_{new}$.

Summarizing the *Fast Approximate Subspace Tracking* (FAST) algorithm presented in [1], specifically equations 14 through 32, we calculate a $k+1 \times c$ matrix, $\boldsymbol{E}$, which when premultiplied by $[\boldsymbol{U}_{old}\ \boldsymbol{q}]$, is the projection of $\boldsymbol{M}_{new}$ onto the columns of $\boldsymbol{U}_{old}$ plus the component of $\boldsymbol{m}_{c+1}$ orthogonal to $\boldsymbol{U}_{old}$. We form the matrix $\boldsymbol{E}$ as follows:

$$\boldsymbol{E} = \left[\begin{array}{cccc|c} \boldsymbol{a}_2 & \boldsymbol{a}_3 & \cdots & \boldsymbol{a}_c & \boldsymbol{a}_{c+1} \\ 0 & 0 & \cdots & 0 & b \end{array}\right], \tag{166}$$

where the $\boldsymbol{a}_j$'s are k element vectors,

$$\boldsymbol{a}_j = \boldsymbol{U}_{old}^H \boldsymbol{m}_j, \qquad j = 2, 3, ..., c+1, \tag{167}$$

b is the norm of the component of $\boldsymbol{m}_{c+1}$ orthogonal to the columns of $\boldsymbol{U}_{old}$, and $\boldsymbol{q}$ is that component normalized. They are computed as follows:

$$\boldsymbol{z} = (\boldsymbol{I} - \boldsymbol{U}_{old}\boldsymbol{U}_{old}^H)\boldsymbol{m}_{c+1}, \quad b = \|\boldsymbol{z}\|, \quad \boldsymbol{q} = \boldsymbol{z}/b. \tag{168}$$

Instead of computing the $O(rc^2)$ SVD [5] of the $r \times c$ matrix $\boldsymbol{A} = [\boldsymbol{U}_{old} \mid \boldsymbol{q}]\boldsymbol{E}$, or the $O((k+1)c^2)$ SVD of the matrix $\boldsymbol{E}$, we compute the $O((k+1)^3)$ SVD of the $(k+1) \times (k+1)$ matrix

$$\boldsymbol{F} = \boldsymbol{E}\boldsymbol{E}^H = \boldsymbol{U}_F \boldsymbol{\Sigma}_F \boldsymbol{U}_F^H, \tag{169}$$

where our estimates of the $k+1$ principal left singular vectors and values of $\boldsymbol{M}_{new}$ are

$$\boldsymbol{U}_{new} = [\boldsymbol{U}_{old} \mid \boldsymbol{q}]\boldsymbol{U}_F \tag{170}$$

and

$$\boldsymbol{\Sigma}_{new} = \sqrt{\boldsymbol{\Sigma}_F}. \tag{171}$$

The original contribution of this paper is the demonstration that the amount of computation in the FAST algorithm can be reduced for the case in which the computation of (eq. 167) can be written as a product of a matrix and a Hankel-like matrix as in (eq. 173). This case arises whenever the matrices $\boldsymbol{M}_{old}$ and $\boldsymbol{M}_{new}$ are formed from a single-channel discrete-time signal [2, 3, 4]. In the last section, the amount of computational reduction is quantified, and a formula (eq. 184) based on the dimensions of the signal matrix (r and c) is given to determine if this method actually reduces computation or not for a given specific interference subspace dimension (k).

3.3 The Use of the FFT

The suggestion made here is that one make use of the fact that the $\boldsymbol{a}_j$'s are a set of convolutions that can be computed using the FFT [5, 6]. To display this as a convolution, we write the ith component of $\boldsymbol{a}_j$ as

$$a_{ij} = \sum_{l=1}^{r} u_{li}^* \, d_{l+j-1}, \qquad j = 2, ...c+1, \tag{172}$$

or in matrix form

$$\boldsymbol{A} = \boldsymbol{U}_{old}^H \boldsymbol{M}_{new} \tag{173}$$

Note that the elements of $\boldsymbol{u}_i$ in (eq. 172) are reversed and conjugated in the correlation sum. This is equivalent to conjugating the DFT of $\boldsymbol{u}_i$.

To efficiently calculate the a_{ij}'s, we pad the columns of $\boldsymbol{U}_{old}$ with zeros, to make them of length N, and compute the DFT's of the columns.

$$\bar{\boldsymbol{U}} \overset{FFT}{\longleftrightarrow} \begin{bmatrix} \boldsymbol{U}_{old} \\ \boldsymbol{0} \end{bmatrix}. \tag{174}$$

We then take the DFT of $\boldsymbol{d}_{new}$

$$\bar{\boldsymbol{d}} \overset{FFT}{\longleftrightarrow} \boldsymbol{d}_{new}. \tag{175}$$

Now, the convolution in (eq. 172) can be performed by the Hadamard product of the conjugate transpose of each column of $\bar{\boldsymbol{U}}$, with the transpose of $\bar{\boldsymbol{d}}$.

$$\bar{\boldsymbol{A}} = \bar{\boldsymbol{U}}^H \odot [\bar{\boldsymbol{d}} \mid \cdots \mid \bar{\boldsymbol{d}}]^T, \tag{176}$$

Finally we take the inverse DFT of the *rows* of $\bar{\boldsymbol{A}}$, whose first c columns are $\boldsymbol{a}_2$ through $\boldsymbol{a}_{c+1}$.

$$\begin{bmatrix} \boldsymbol{a}_2 & \boldsymbol{a}_3 & \cdots & \boldsymbol{a}_{c+1} & \cdots \end{bmatrix} \overset{IFFT}{\longleftrightarrow} \bar{\boldsymbol{A}}. \tag{177}$$

3.4 Operations Count

The two dominant operations in the algorithm are the calculations of the $\boldsymbol{a}_j$'s (eq. 167) for small signal subspace dimensions, and the SVD of F (eq. 169) for large dimensions. This method addresses the calculation of the $\boldsymbol{a}_j$'s.

For the comparisons, we will estimate the number of floating point operations (flops) for each method. We will assume that both real addition and real multiplication require one flop, while complex addition requires two flops and complex multiplication requires six flops.

The calculation of the $\boldsymbol{a}_j$'s using equation (167) requires

$$N_{Adir} = kN_{DIR} \text{ flops}, \tag{178}$$

where

$$N_{DIR} = 2rc \text{ flops} \tag{179}$$

when the data are real, and

$$N_{DIR} = 8rc \text{ flops} \tag{180}$$

when the data are complex. Remember that r and c are the dimensions of $\boldsymbol{M}_{new}$, and k is the signal subspace dimension.

When calculating the $\boldsymbol{a}_j$'s using equations (174) through (177) we must first come up with an approximation of the flop count of an N point FFT. We take, as an approximation of the number of flops for a simple radix 2 FFT,

$$N_{FFT} = 5/2N\log_2(N) - N/2 \text{ flops} \tag{181}$$

when the data are real, and

$$N_{FFT} = 5N\log_2(N) - 3N \text{ flops} \tag{182}$$

when the data are complex [7]. A radix 4 algorithm, with a radix 2 step for N equal to an odd power of 2, takes 25% fewer flops. When N is not a power of 2 and N has large prime factors the FFT can take more flops. There are $2k+1$ FFT's, so, adding to the above the $6Nk$ flops for the complex multiplication of equation (176), we get

$$N_{Afft} = (2k+1)N_{FFT} + 6Nk \text{ flops.} \tag{183}$$

For any r and c, k can range from 0 to k_{max}, where $k_{max} = \min(r, c)$. The transition signal subspace dimension, k_{trans}, where all values of k above that value take less flops using the FFT method, and all values of k below that value take less flops using the direct multiplication method is

$$k_{trans} = \frac{N_{FFT}}{N_{DIR} - 2N_{FFT} - 6N}. \tag{184}$$

When $k_{trans} < 0$ the direct multiplication method is more efficient for all k.

To give an idea of how k_{trans} evaluates for different r and c, we use the radix 2 FFT flop count and assume complex data. Figure 9 shows that there is a very steep transition from where all k are more efficiently calculated using the direct multiplication method to where all k are more efficiently calculated using the FFT method. Given values for r and c it is easy to determine which method to use.

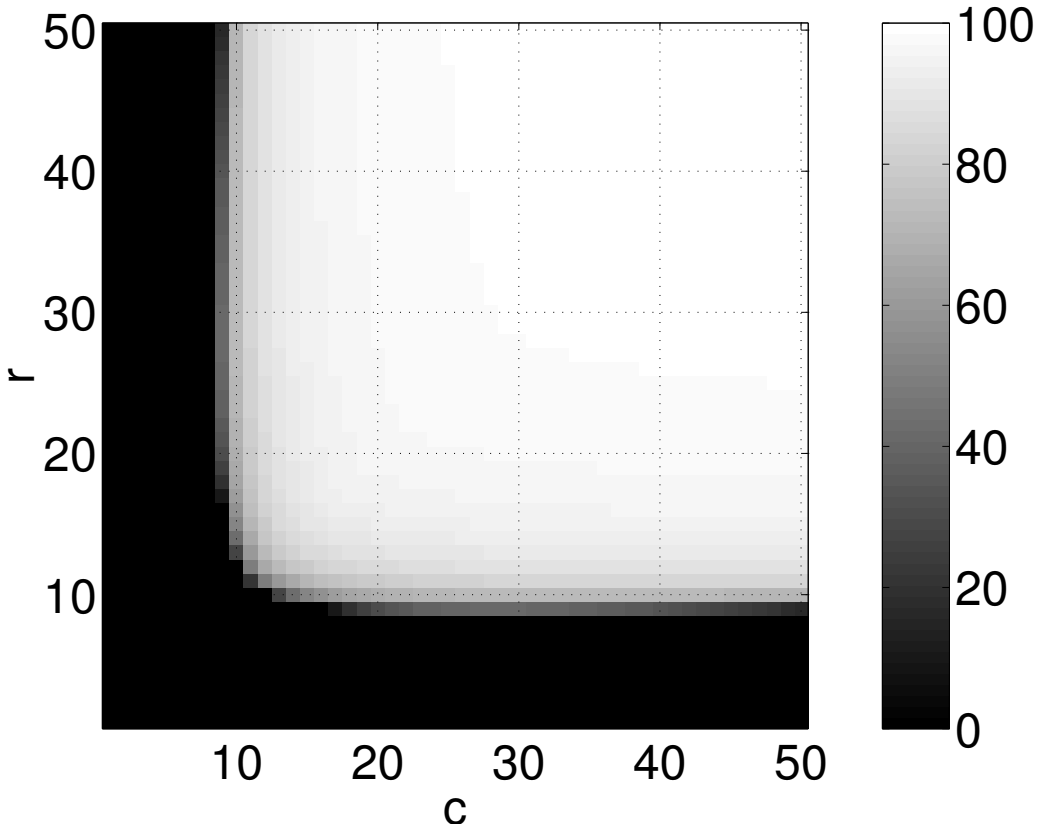


Figure 9. Percentage of all k's calculated more efficiently using the FFT method for r and c from 1 to 50.

Figure 10 shows the percentage of signal subspace dimensions, k, where the FFT method is more efficient than the direct multiplication method. In this figure, $r = c$, and both real and complex data are shown. It can be seen that for complex matrices with dimensions greater than 15 (which is often the case) and real matrices with dimensions greater than 40, one would probably want to use the FFT method.

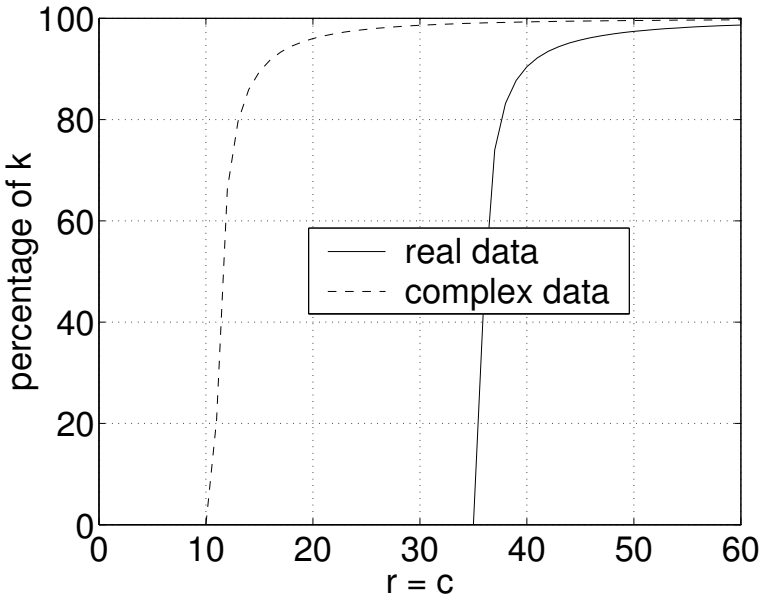


Figure 10. Percentage of all k's calculated more efficiently using the FFT method for $r = c$ from 1 to 60 and both real and complex data.

3.5 Summary

We have established that, for the above signal tracking method, where the signal matrix is a Hankel-like matrix, the FFT method may be superior for large r and c. The above formulas should enable one to evaluate which method is more efficient for any given parameters of the problem.

List of References

[1] E. C. Real, D. W. Tufts, and J. W. Cooley, "Two algorithms for fast approximate subspace tracking," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 47, no. 7, pp. 1036–1045, July 1999.

[2] D. W. Tufts, R. Kumaresan, and I. Kirsteins, "Data adaptive signal estimation by singular value decomposition of data matrix," *Proceedings of the IEEE*, vol. 70, no. 6, pp. 684–685, June 1982.

[3] D. W. Tufts and R. Kumaresan, "Singular value decomposition and improved frequency estimation using linear prediction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-30, no. 4, pp. 671–675, Aug. 1982.

[4] L. L. Scharf and D. W. Tufts, "Rank reduction for modeling stationary signals," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 3, pp. 350–355, Mar. 1987.

[5] G. H. Golub and C. F. van Loan, *Matrix Computations, 3rd ed.* Baltimore, MD: Johns Hopkins Univ. Press, 1996.

[6] T. Kailath and A. H. Sayed, *Fast Reliable Algorithms for Matrices with Structure*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.

[7] J. W. Cooley, P. A. Lewis, and P. D. Welch, "The Fast Fourier Transform algorithm and its applications," IBM Watson Research Center, Yorktown Heights, NY, Tech. Rep. RC-1743, Feb. 1967.

MANUSCRIPT 4

Detection and Estimation in Non-Stationary Environments

4.1 Abstract

In this manuscript, we describe a matrix based method for estimating both a signal subspace dimension as well as signal parameters, when one has an array of sensors, multiple exponential signals, and significant signal changes after a small number of snapshots. This method combines the technique of creating Hankel or Toeplitz matrices from single-channel data with methods for sensor-array processing using multiple array snapshots.

4.2 Introduction

In this manuscript, we describe a matrix based method for estimating a signal subspace dimension while controlling the probability of false alarm as well as a method for estimating parameters of that subspace. A common scenario where this method applies is when the data comes from an array of sensors, where each array snapshot consists of multiple exponential signals, but significant changes occur in the exponential signals after a small number of snapshots.

In estimation, these exponentials are signals which have parameter values, such as arrival angles, which we wish to estimate. In detection, these exponentials are components of interference which we temporarily treat as "signals" to be enhanced, prior to subtraction. In both cases, reduced-rank approximation to a data matrix is used to improve the signal-to-noise ratio of the exponential components, prior to subsequent signal processing.

We discuss a data matrix structure, which we call the *block Hankel* structure, that combines the benefits of creating a Hankel matrix for single-channel data with the advantages of multiple channels without changing the signal subspace

dimension.

Next, we present a method for estimating the dimension of the signal subspace while controlling the probability of false alarms. If we estimate that the dimension of the signal subspace is larger than the correct dimension, we say that a false alarm has occurred. This method is an extension of the method of Tufts and Shah [1] which applies to Hankel matrices. We then introduce an approximation to this matrix rank tracking method that reduces the computation significantly, while maintaining performance.

A major motivation for us is widening the applicability of the FAST algorithm [2, 3] for subspace tracking. Rank tracking, implemented using the tests of Frobenius-norm "energy" of subspace matrices is described in section 4.4 below. This is an important part of FAST. However, until now, the rank-tracking in FAST could not be applied to *block Hankel* structure matrices.

Finally, we present some results of applying this method to some simulated sonar array data which was generated by Norman Owsley. Here we estimate the number of sinusoids, their amplitudes, and their spatial frequencies for each snapshot.

4.3 Constructing a Block Hankel Matrix

It is well known [4, 5, 6] that a length N single-channel signal vector $\boldsymbol{s}_t$, which is a linear combination of k complex exponentials can be made into an $r_H \times c_H$ Hankel or Toeplitz matrix which will have rank k, if $\min(r_H, c_H) \geq k$. The vector $\boldsymbol{s}_t$ can be written as

$$\boldsymbol{s}_t = \sum_{l=1}^{k} c_{l,t} \boldsymbol{z}_l, \tag{185}$$

where each discrete exponential signal has the form

$$\boldsymbol{z}_l = [1 \;\; Z_l^1 \;\; Z_l^2 \;\; \cdots \;\; Z_l^{N-1}]^T, \tag{186}$$

in which Z_l is a complex number and the $c_{l,t}$ are the complex scale factors. The creation of a Hankel matrix is shown pictorially in Figure 11. Note that $N = r_H + c_H - 1$.

When we have c different signal vectors, $\boldsymbol{s}_1, \boldsymbol{s}_2 \cdots \boldsymbol{s}_c$, with kc different complex scale factors $c_{l,t}$, but the same k exponentials, $\boldsymbol{z}_l$, we can create a *block Hankel* matrix by forming an $r_H \times c_H$ Hankel matrix from each $N \times 1$ signal vector, $\boldsymbol{s}_t$, then concatenating them together to form an $r_H \times c_H c$ matrix which will have rank k, if $\min(r_H, c_H c) \geq k$.

Given an $N \times c$ data matrix M, consisting of c snapshots of signal plus noise,

$$M = [\boldsymbol{s}_1 \ \boldsymbol{s}_2 \ \cdots \ \boldsymbol{s}_c] + [\boldsymbol{n}_1 \ \boldsymbol{n}_2 \ \cdots \ \boldsymbol{n}_c], \tag{187}$$

we can create an $r_H \times c_H c$ *block Hankel* matrix B. This is shown pictorially in Figure 12. We often refer to a column of the original data matrix as a snapshot.

The noise component of M will increase its dimension when we create the Hankel blocks, and should continue to fill the full vector space of M. As an example of how the *block Hankel* structure benefits us, if we have four snapshots of length 39 which contain eight complex exponentials, we can create four 32×8 Hankel blocks, which will give us a 32×32 *block Hankel* matrix in which the signal component is contained in an eight dimensional subspace, but the noise will span

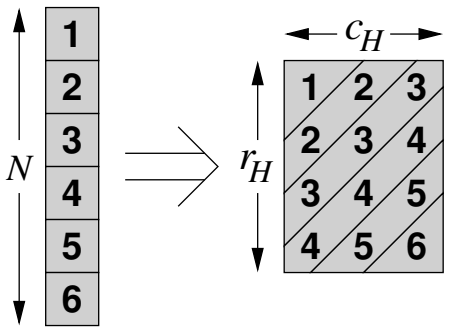


Figure 11. Creating a Hankel matrix from a signal vector

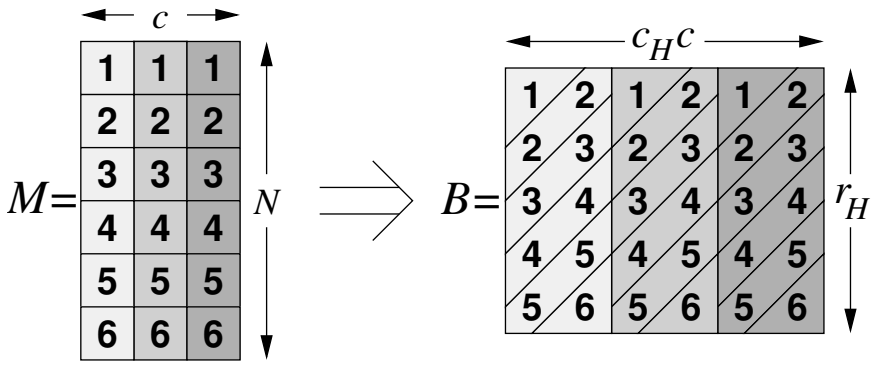


Figure 12. Creating a Block Hankel matrix from multiple snapshots

the full 32 dimensional vector space.

It should also be noted that when $Z_l = e^{jw_l}$ in (186), we can create a forward-backward matrix [7] where the backward matrix is created by conjugating and reversing the elements of M along the columns. The results in this manuscript are only shown for the forward matrix, but can be easily extended to contain both the forward and backward matrices.

4.4 Estimating the Signal Subspace Rank

To estimate the rank of the signal subspace, we take the SVD of $B = U\Sigma V^H$, and define the energy in the subspace which is orthogonal to the hypothesized signal subspace

$$S_{k+1} = \sum_{l=k+1}^{r_H} \sigma_l^2 = \|(I - U_k U_k^H)B\|_F^2 \tag{188}$$

where S_{k+1} is the sum of the squares of the singular values of B less the k largest. In (188) σ_l^2 is the square of the lth largest singular value of B, and U_k is a matrix of the k leftmost columns of U.

Using the SVD of the matrix B, we ask questions based on current hypotheses about the rank of the signal subspace. The zero$^{\text{th}}$ hypothesis, H_0, is that the rank of the signal subspace, the signal portion of the matrix B, is at least zero. If the signal portion of the matrix B has exactly rank zero, then there is no signal and

the matrix B consists entirely of noise values. The k^{th} hypothesis, H_k, is that the rank of the signal portion of the matrix B is at least k.

The question that we ask at the k^{th} stage (if we get that far) is "Given H_k, that we have found out that the signal rank is at least k, can we now say that H_{k+1} is true?" To do this, we test whether or not the sum S_{k+1}, the energy in the orthogonal subspace, is greater than a prescribed threshold value T_k. If $s_{k+1} < T_k$, we say that the signal rank is k and stop our tests. If $S_{k+1} > T_k$ we say that H_{k+1} is true and continue our tests. The behavior of this sequence of tests is controlled by choosing each threshold value so that the associated probability of false alarm is a value α which we choose.

We choose a false alarm probability, α, and compute the threshold values, T_k, for each k

$$P(S_{k+1} > T_k | \bar{H}_{k+1}) = \alpha, \qquad 0 \leq k \leq r_H \tag{189}$$

where $\bar{H}_{k+1}$ is the complimentary hypothesis that H_{k+1} is not true, and the value of S_{k+1} is produced only by noise. Note that α should be the same for all k.

Finally, we find the largest k such that S_{k+1} is greater than T_k, and our rank is that k. This is a simple iterative step that is trivial to implement in practice.

These steps for estimating the rank apply for any matrix, structured or not, because the original matrix M is a *block Hankel* matrix with $r_H = N$ and $c_H = 1$, while a Hankel matrix is a *block Hankel* matrix with $c = 1$ and $c_H > 1$.

4.5 Calculating the Threshold Values

The threshold values are chosen to control the probability of false alarm at each stage. Therefore, the pertinent probability density is that of the noise alone in the orthogonal subspace.

A method for calculating the thresholds T_k for an unstructured matrix, such as M, is presented in [8], and a method for calculating the thresholds in the Hankel

case, which is easily extended to the *Block Hankel* case, is presented in [1].

The difficulty with the method in [1] is that it requires the partial fraction expansion of a polynomial with root multiplicity of $2c$. For the case of a Hankel matrix this is not a big problem because $c = 1$, but for the *Block Hankel* case this not only requires a lot of computation, but also generally requires variable precision arithmetic.

Here we present a method to approximate the threshold values which can easily be implemented in a practical system. They are only a function of α, σ^2, and the matrix dimensions, r_H, c_H, and c. The values are compared with the results using the extension of the method in [1] as well as experimental results.

We now assume that the noise is distributed complex normal, $\boldsymbol{n}_t \sim \mathcal{CN}(0, I\sigma^2)$, with zero mean and variance σ^2. For the case H_0 (no signal present) the expected value, μ_B, and variance, σ_B^2, of the squared Frobenius norm of B are

$$\mu_B = E\left[\|B\|_F^2\right] = \sigma^2 r_m c_m c \tag{190}$$

and

$$\sigma_B^2 = \mathrm{Var}\left(\|B\|_F^2\right) = \sigma^4 c(d r_m^2 + 2\sum_{i=1}^{r_m - 1} i^2), \tag{191}$$

where $r_m = \min(r_H, c_H)$ is the smaller dimension of a single Hankel block, $c_m = \max(r_H, c_H)$ is the longer dimension of a single block, and $d = |r_H - c_H| + 1$ is the number of full diagonals in a single block.

The distribution of $\|B\|_F^2$ is Chi-Square mixture, which is approximately a scaled Chi-Square with n degrees of freedom and scale factor $1/s_B$. Therefore, because we know the formula for the mean and variance of any Chi-Square variable, we can say

$$n = E\left[\frac{\|B\|_F^2}{s_B}\right] = \frac{1}{s_B} E\left[\|B\|_F^2\right] \tag{192}$$

and

$$2n = \mathrm{Var}\left(\frac{\|B\|_F^2}{s_B}\right) = \frac{1}{s_B^2}\mathrm{Var}\left(\|B\|_F^2\right). \tag{193}$$

Combining equations (190), (191), (192), and (193), rearranging some terms, and solving for n and s_B we get

$$n = \frac{2\mu_B^2}{\sigma_B^2} = \frac{6cc_m^2}{3c_m - (r_m - 1/r_m)} \tag{194}$$

and

$$s_B = \frac{\sigma_B^2}{2\mu_B} = \sigma^2 \frac{r_m c_m c}{n}. \tag{195}$$

It should be noted that n will generally not be an integer, but that is not a problem because the Chi-Square distribution can be evaluated for all real n.

For a given value of α, we can find T_0/s_B by evaluating the quantile (the inverse cumulative distribution function) of the Chi-Square distribution at $1-\alpha$.

$$\frac{T_0}{s_B} = F_n^{-1}(1-\alpha) \tag{196}$$

Since the quantile is only a function of α and n, and n depends only on the matrix dimensions r_m, c_m, and c, we can calculate T_0/s_B before we know the variance of the noise, σ^2. If we then define

$$\hat{T}_0 = \sigma^2 \frac{T_0}{s_B}, \tag{197}$$

which is essentially T_0 with the noise variance in s_B canceled out, then when we do get our estimate of the noise variance we can easily determine T_0 as

$$T_0 = \frac{\hat{T}_0}{\sigma^2}. \tag{198}$$

In Figure 13, we show how well the Chi-Square approximation compares to the actual distribution which is a Chi-Square mixture.

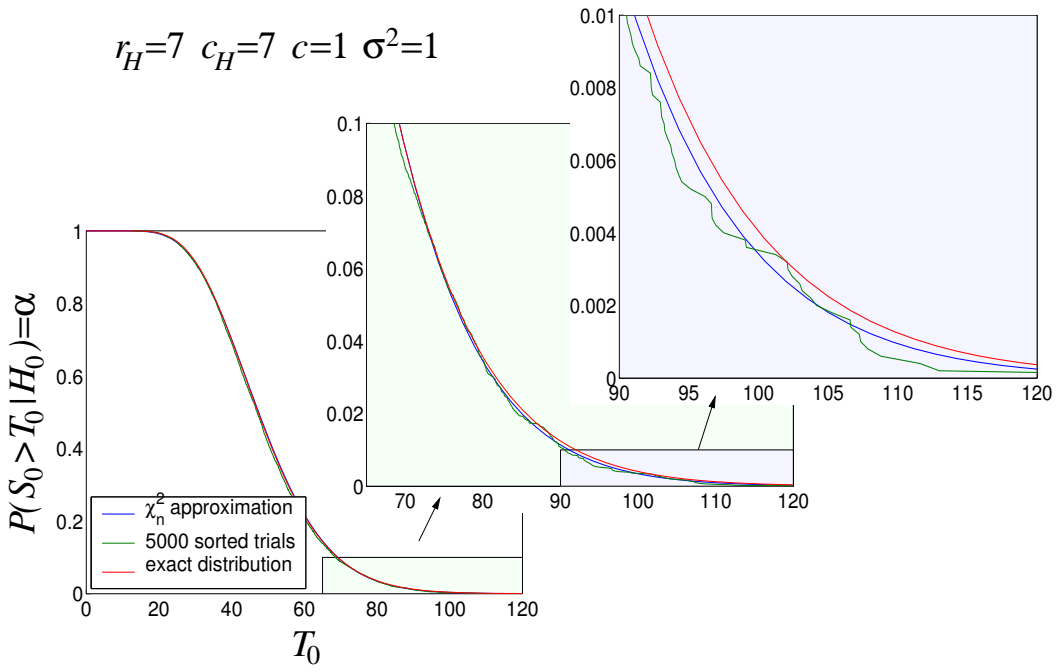


Figure 13. False Alarm Probability vs. Threshold

4.6 Evaluating the Other Thresholds

Now that we have a method to calculate T_0, we need to be able to calculate the other thresholds T_k, for $k = 1, \cdots, r_H$. Here, we assume that if there are signals present in the data, the SNR is assumed to be above threshold [9]. That is, the probability of subspace swap is negligibly small and the signal singular vectors are independent of the noise.

We assume that, to a good approximation, the mean and variance of the energy in the orthogonal subspace do not depend on the choice of signal subspace. Therefore, for convenience we replace U_k by the first k canonical vectors,

$$\hat{U}_k = [\, \boldsymbol{e}_1,\ \boldsymbol{e}_2, \cdots, \boldsymbol{e}_k \,], \tag{199}$$

where the kth canonical vector $\boldsymbol{e}_k$ is a length r_H column vector consisting of all zeros except a single one in the kth position,

$$\boldsymbol{e}_k = [\, \underbrace{0, \cdots, 0}_{k-1}, 1, \underbrace{0, \cdots, 0}_{r_H - k} \,]^T, \tag{200}$$

we can use the method from Section 4.5 for calculating T_0 to calculate T_k by replacing r_H by $r_H - k$.

We see that when we take the product $(I - \hat{U}\hat{U}^H)B$, we zero out the first k rows of B but leave the rest of the matrix unchanged. This means that if hypothesis H_k applies, we can use our mean and variance calculations from the previous section along with our Chi-Square approximation. The mean estimate using $\hat{U}_k$ will be identical to the estimate using U_k, but the variance will not be correct because $\|(I - UU^H)B\|_F^2$ will actually be a Chi-Square mixture plus a Gaussian product mixture.

The reason for this approximation is to permit the thresholds to be calculated independently of the data.

4.7 The Data

In this section, we present some results using the techniques introduced in this manuscript on simulated data. We make the following assumptions about the data used in this section.

Each length 48 array snapshot $\boldsymbol{m}_t$, is a sum of k scaled complex sinusoids with fixed frequencies f_k, and random complex scale factors $c_{k,t} = A_k e^{j\psi_k}$, plus complex white noise $\boldsymbol{n}_t$

$$\boldsymbol{m}_t = \boldsymbol{n}_t + \sum_{l=1}^{k} c_{k,t}\boldsymbol{z}_k \tag{201}$$

where from (186), $Z_k = e^{-j2\pi f_k f_s}$ with $f_s = 0.4$, and the random components have distributions

$$\boldsymbol{n}_t \sim \mathcal{CN}(0, I\sigma^2), \qquad A_k \sim \mathcal{N}(0, \sigma_k^2), \qquad \psi_k \sim \mathcal{U}(0, 2\pi). \tag{202}$$

All three of $\boldsymbol{n}_t$, A_k and ψ_k are different for each t.

Because we know exactly how the simulated data were generated, we also know that these assumptions are simplifications of the actual data, and do not

truly reflect the far more complex model used for generating the data. In actuality, the $\boldsymbol{z}_k$s are not truly sinusoidal (which is why we didn't use forward-backward *block Hankel* matrices), the f_ks are slowly changing between snapshots at different rates, and the $c_{k,t}$ have a much more complicated distribution.

The steps that we use to come up with the results in this section are as follow.

- Determine c, the number of sequential snapshots to use. This will depend on the stationarity of the signal subspace.
- Determine r_H and c_H, the dimensions of the Hankel blocks. This will depend on the rank of the signal subspace as well as other factors related to the method of parameter estimation that is used.
- Determine α, the probability of false alarm, then calculate the thresholds T_k or $\hat{T}_k$ for each k.
- Create the *block Hankel* matrix B, and take its SVD.
- Estimate k, the signal subspace rank by comparing the sums of the squares of singular values of B to the thresholds.
- Estimate the possible target azimuths.
- Find the k azimuths corresponding to the signal subspace, and determine their signal level.

To estimate the possible target azimuths, we take the polynomial roots of the r_Hth left singular vector which will be orthogonal to the signal subspace. We know that it will have zeros corresponding to the frequencies of the sinusoids in the signal subspace [5] (as well as many other zeros).

To determine which k of the $r_H - 1$ possible azimuths correspond to the k sinusoids, we beamform the k largest left singular vectors toward all of the possible azimuths, then pick the azimuth which has the largest beamformed value for a given singular vector. Once we have picked an azimuth which corresponds to a singular vector, we say the energy at that azimuth is the singular value which goes with that singular vector.

Figure 14 shows the rank estimates for 1800 snapshots using a *block Hankel* matrix with dimensions $c = 8$, $r_H = 33$, and $c_H = 16$. These are the number of azimuth estimates that are plotted in Fig. 15 and Fig. 18.

Figure 15 shows the cosine of azimuth estimates using eight sequential snapshots and a *block Hankel* structure with $c = 8$, $r_H = 33$, and $c_H = 16$. Fig. 17 shows the cosine of azimuth estimates using the same eight sequential snapshots and no matrix structure with $c = 8$, $r_H = 48$, and $c_H = 1$. Fig. 16 shows the cosine of azimuth estimates using 24 sequential snapshots and no matrix structure with $c = 24$, $r_H = 48$, and $c_H = 1$.

The two tracks of most interest are the one that is leftmost between 500 and 1600 and the one that is rightmost between 800 and 1400 in Fig. 15. These two tracks are about five orders of magnitude below the stronger tracks and very near the noise level. They do not even show up Fig. 17 and are not very clear in Fig. 16 which uses three times the amount of data.

Figure 18 is the same as Fig. 15 but with target strength indicated by a color. The colorbar in the figure shows the strength of the target in decibels.

List of References

[1] D. W. Tufts and A. A. Shah, "Rank determination in time-series analysis," in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP)*, Apr. 1994, pp. IV–21–IV–24.

[2] E. C. Real, D. W. Tufts, and J. W. Cooley, "Two algorithms for fast approximate subspace tracking," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 47, no. 7, pp. 1036–1045, July 1999.

[3] J. W. Cooley, T. M. Toolan, and D. W. Tufts, "A subspace tracking algorithm using the Fast Fourier Transform," *IEEE Signal Processing Letters*, vol. 11, no. 1, pp. 30–32, Jan. 2004.

[4] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 561–580, Apr. 1975.

[5] R. Kumaresan and D. W. Tufts, "Estimating the parameters of exponentially damped sinusoids and pole-zero modeling in noise," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 30, no. 6, pp. 833–840, Dec. 1982.

[6] G. H. Golub and C. F. van Loan, *Matrix Computations, 3rd ed.* Baltimore, MD: Johns Hopkins Univ. Press, 1996.

[7] D. W. Tufts and R. Kumaresan, "Estimation of frequencies of multiple sinusoids: Making linear prediction perform like maximum likelihood," *Proceedings of the IEEE*, vol. 70, no. 9, pp. 975–989, Sept. 1982.

[8] A. A. Shah and D. W. Tufts, "Determination of the dimension of a signal subspace from short data records," *IEEE Transactions on Signal Processing*, vol. 42, no. 9, pp. 2531–2535, Sept. 1994.

[9] D. W. Tufts, A. C. Kot, and R. J. Vaccaro, "The threshold analysis of SVD-based algorithms," in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP)*, Apr. 1988, pp. 2416–2419.

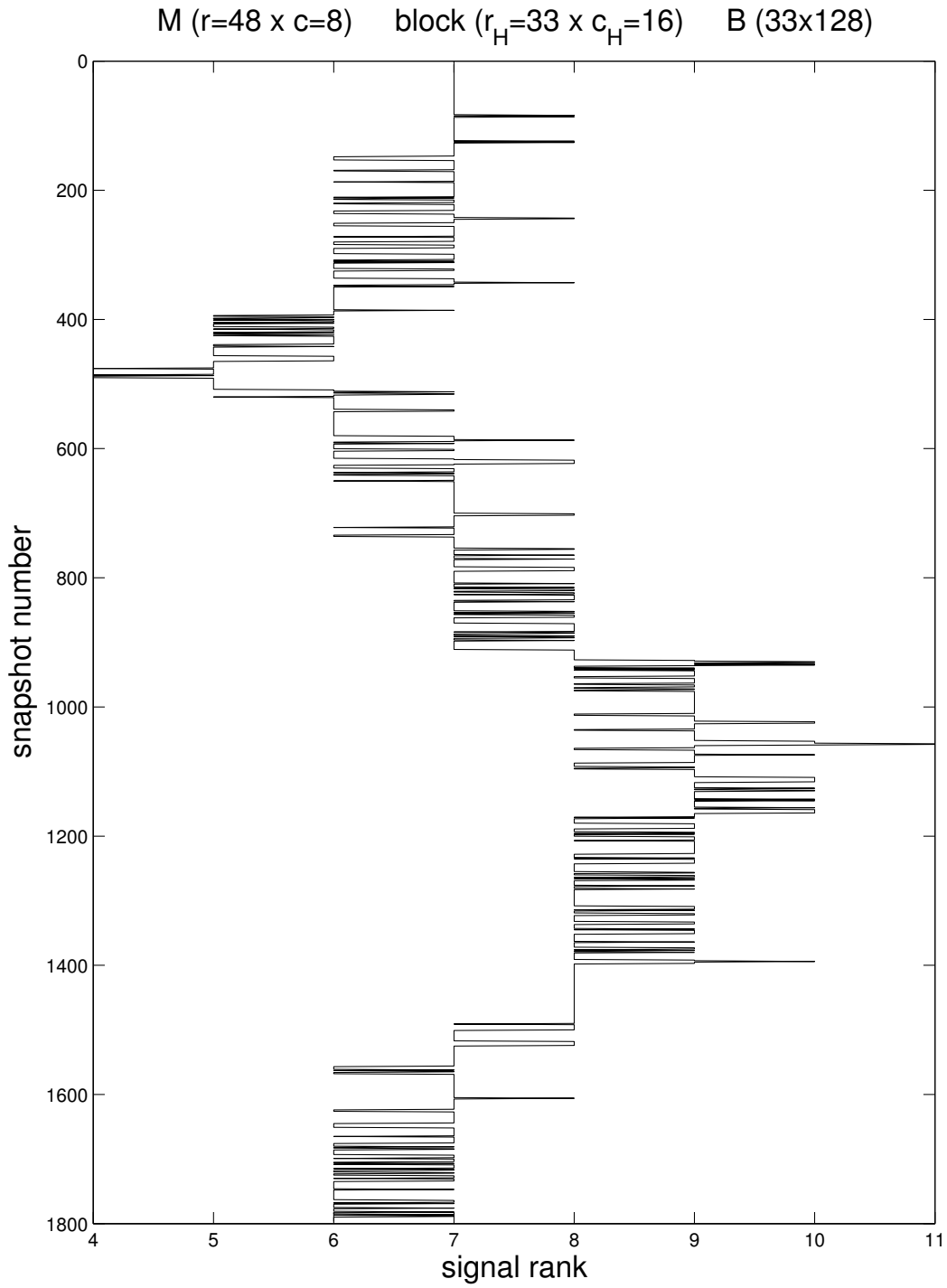


Figure 14. Rank estimation for block Hankel matrix structure with eight sequential snapshots.

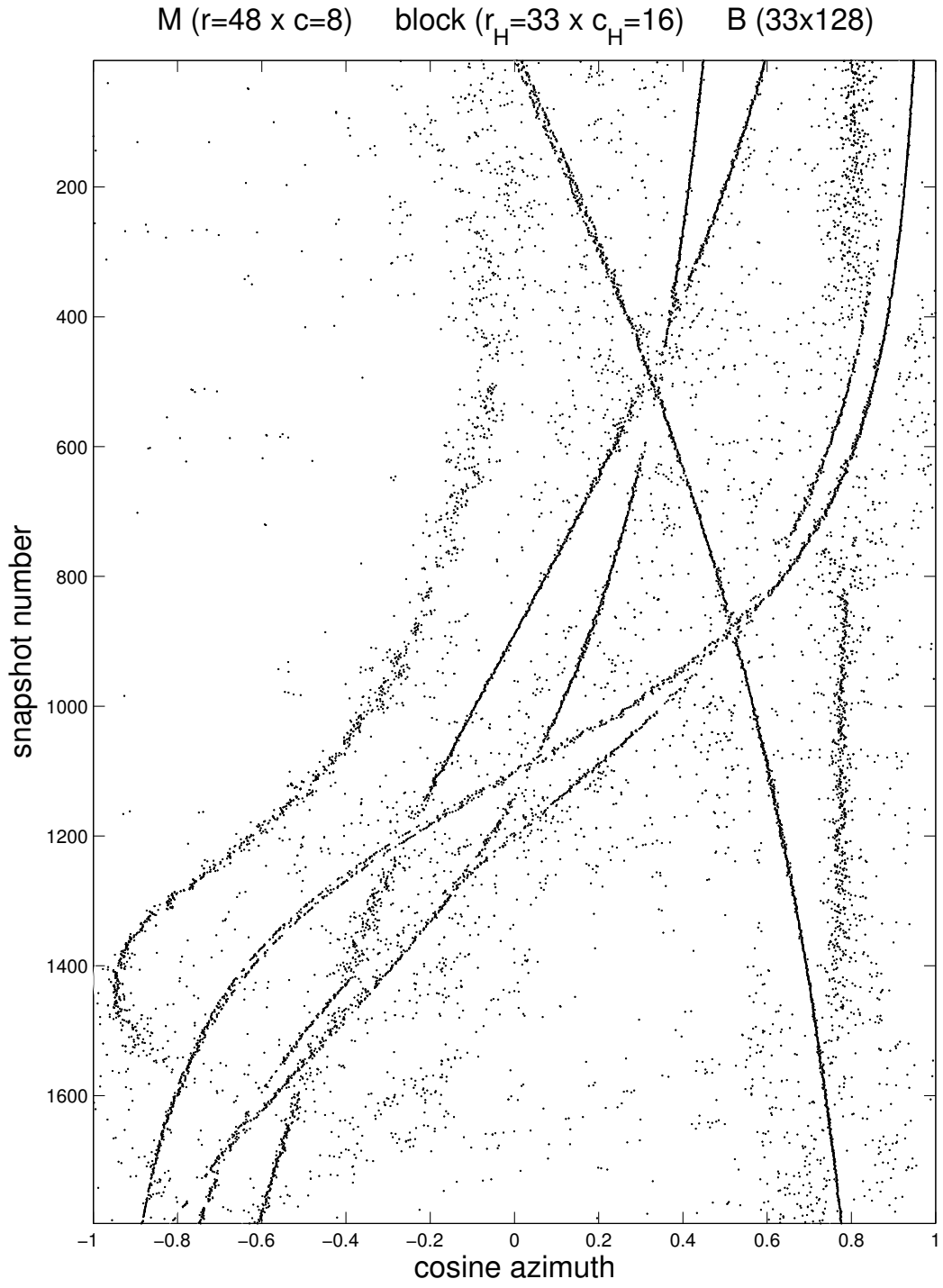


Figure 15. Cosine of the azimuth of the k strongest sinusoids using eight sequential snapshots and block Hankel matrix structure.

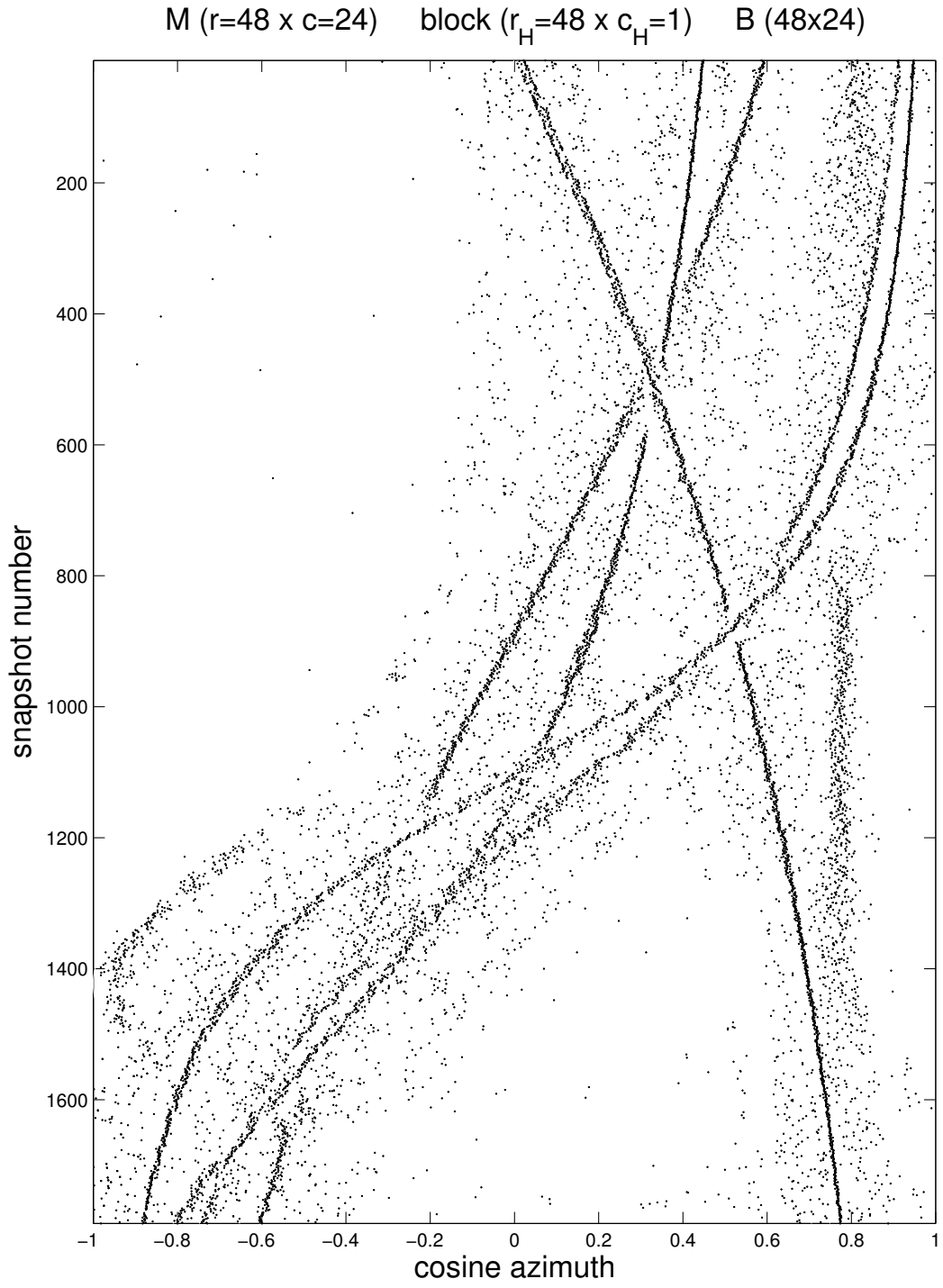


Figure 16. Cosine of the azimuth of the k strongest sinusoids using 24 sequential snapshots and no matrix structure.

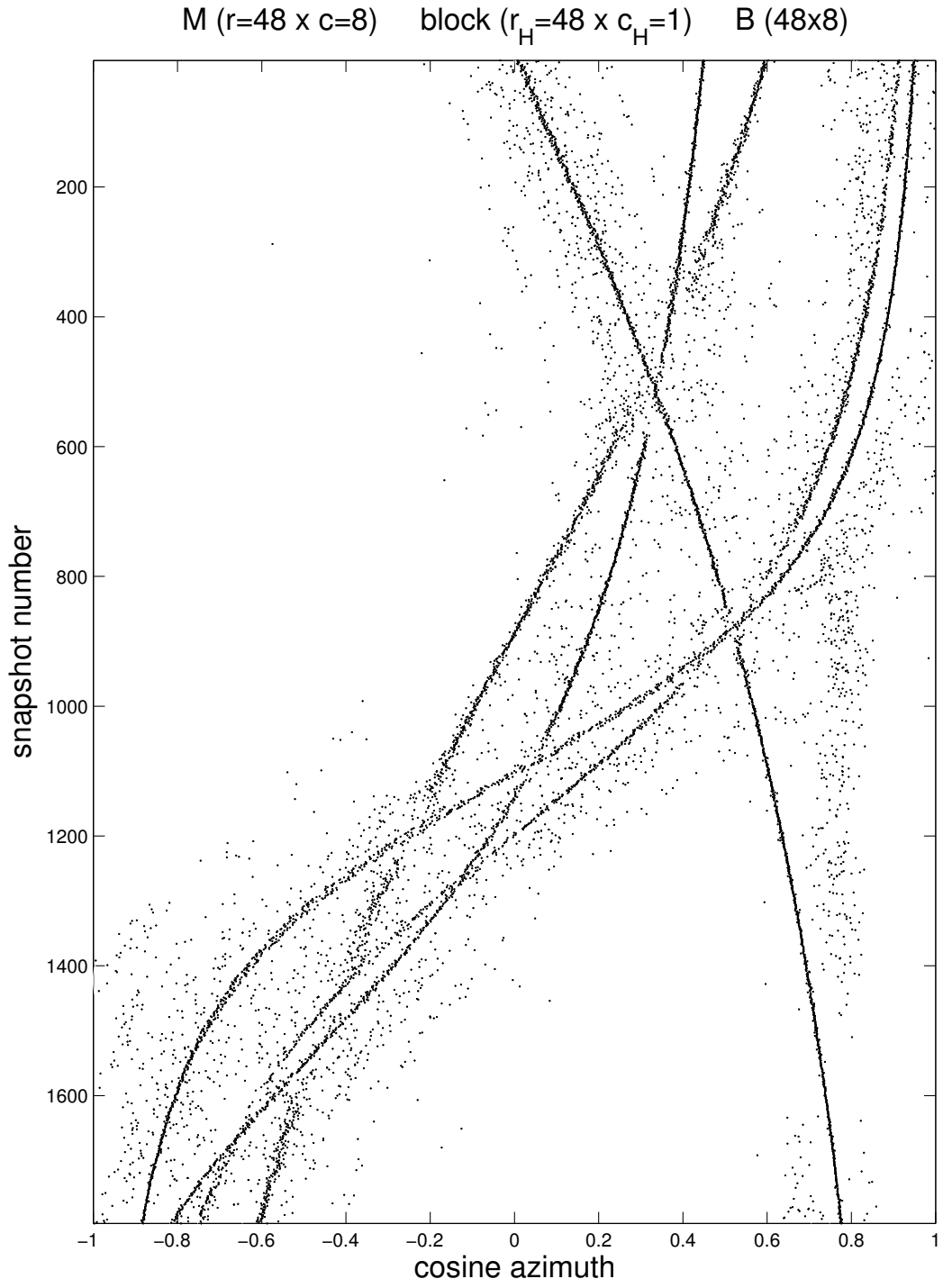


Figure 17. Cosine of the azimuth of the k strongest sinusoids using eight sequential snapshots and no matrix structure.

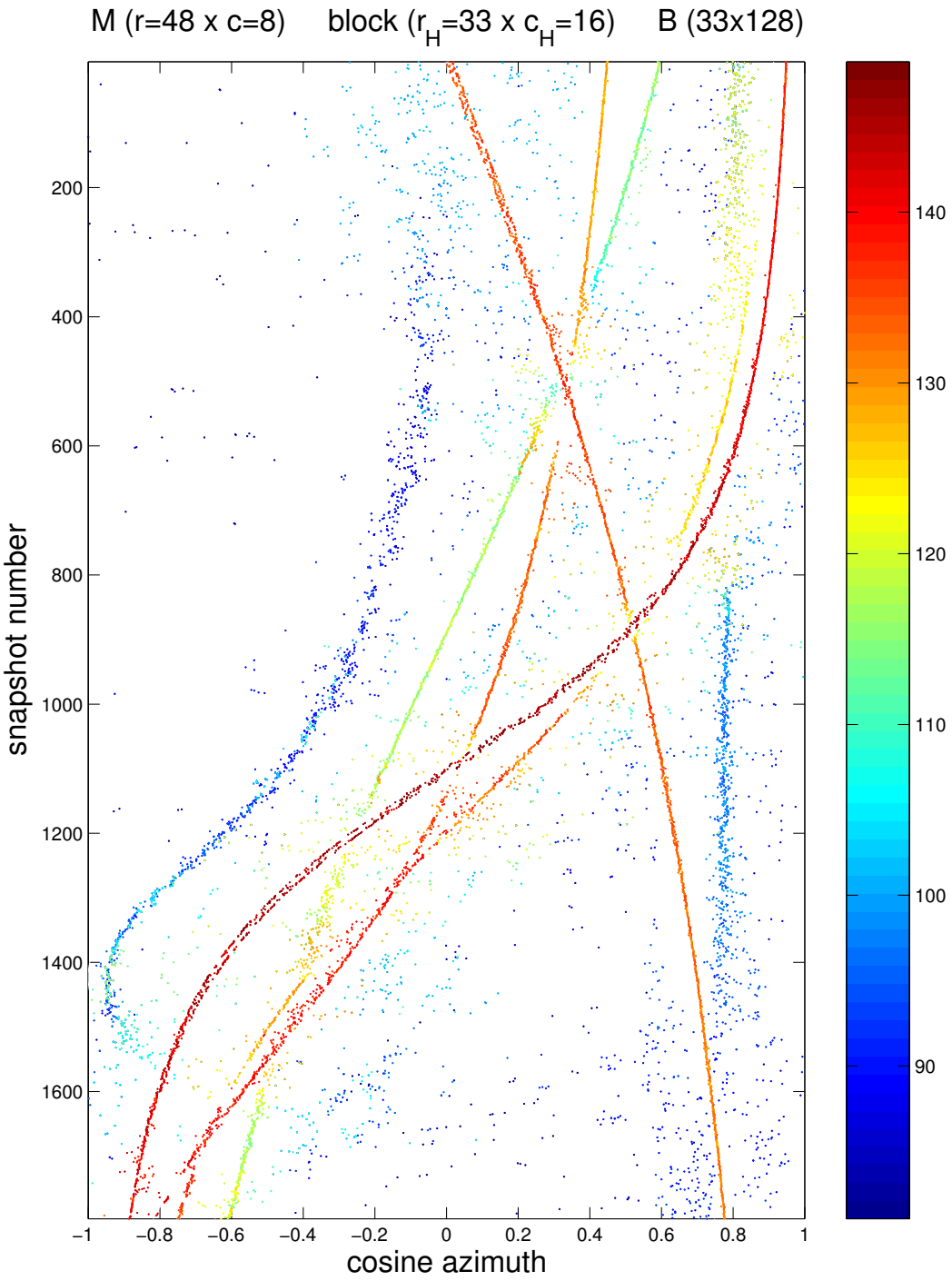


Figure 18. Cosine of the azimuth of the k strongest sinusoids using eight sequential snapshots and block Hankel matrix structure with color indicating target strength.

BIBLIOGRAPHY

Bunch, J. R., Nielsen, C. P., and Sorensen, D. C., "Rank-one modification of the symmetric eigenproblem," *Numerische Mathematik*, vol. 31, pp. 31–48, 1978.

Cooley, J. W., Lewis, P. A., and Welch, P. D., "The Fast Fourier Transform algorithm and its applications," IBM Watson Research Center, Yorktown Heights, NY, Tech. Rep. RC-1743, Feb. 1967.

Cooley, J. W., Toolan, T. M., and Tufts, D. W., "A subspace tracking algorithm using the Fast Fourier Transform," *IEEE Signal Processing Letters*, vol. 11, no. 1, pp. 30–32, Jan. 2004.

Cooley, J. W. and Tukey, J. W., "An algorithm for the machine computation of complex Fourier series," *Mathematics of Computation*, vol. 19, pp. 297–301, Apr. 1965.

Demmel, J. W., *Applied Numerical Linear Algebra*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1997.

Gantmacher, F. R., *Matrix Theory*. New York, NY: Chelsea Publishing Company, 1959.

Golub, G. H., "Some modified matrix eigenvalue problems," *SIAM Review*, vol. 15, no. 2, pp. 318–334, Apr. 1973.

Golub, G. H. and van Loan, C. F., *Matrix Computations, 3rd ed.* Baltimore, MD: Johns Hopkins Univ. Press, 1996.

Gu, M. and Eisenstat, S. C., "A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem," *SIAM Journal on Matrix Analysis and Applications*, vol. 15, no. 4, pp. 1266–1276, Oct. 1994.

Gu, M. and Eisenstat, S. C., "Downdating the singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 16, no. 3, pp. 793–810, July 1995.

Kailath, T. and Sayed, A. H., *Fast Reliable Algorithms for Matrices with Structure*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.

Karasalo, I., "Estimating the covariance matrix by signal subspace averaging," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, no. 1, pp. 8–12, Feb. 1986.

Kumaresan, R. and Tufts, D. W., "Estimating the parameters of exponentially damped sinusoids and pole-zero modeling in noise," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 30, no. 6, pp. 833–840, Dec. 1982.

Makhoul, J., "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 561–580, Apr. 1975.

Real, E. C., Tufts, D. W., and Cooley, J. W., "Two algorithms for fast approximate subspace tracking," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 47, no. 7, pp. 1036–1045, July 1999.

Real, E. C., Yannone, R. M., and Tufts, D. W., "Comparison of two methods for multispectral 3-D detection of single pixel features in strong textured clutter," in *Proceedings Image and Multidimensional Digital Signal Processing Conference (IMDSP), Alpbach, Austria*, July 1998.

Scharf, L. L. and Tufts, D. W., "Rank reduction for modeling stationary signals," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 3, pp. 350–355, Mar. 1987.

Shah, A. A. and Tufts, D. W., "Determination of the dimension of a signal subspace from short data records," *IEEE Transactions on Signal Processing*, vol. 42, no. 9, pp. 2531–2535, Sept. 1994.

Toolan, T. M. and Tufts, D. W., "Detection and estimation in non-stationary environments," in *Proceedings IEEE Asilomar Conference on Signals, Systems & Computers*, Nov. 2003, pp. 797–801.

Toolan, T. M. and Tufts, D. W., "Improved fast adaptive subspace tracking," in *Proceedings Thirteenth Adaptive Sensor Array Processing Workshop (ASAP05)*, MIT Lincoln Laboratory, Lexington MA, June 2005.

Tufts, D. W., "Keynote address," in *MIT Lincoln Laboratory Adaptive Array Signal Processing (ASAP) Conference, Lexington, MA*, Mar. 2001.

Tufts, D. W., Kot, A. C., and Vaccaro, R. J., "The threshold analysis of SVD-based algorithms," in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP)*, Apr. 1988, pp. 2416–2419.

Tufts, D. W. and Kumaresan, R., "Estimation of frequencies of multiple sinusoids: Making linear prediction perform like maximum likelihood," *Proceedings of the IEEE*, vol. 70, no. 9, pp. 975–989, Sept. 1982.

Tufts, D. W. and Kumaresan, R., "Singular value decomposition and improved frequency estimation using linear prediction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-30, no. 4, pp. 671–675, Aug. 1982.

Tufts, D. W., Kumaresan, R., and Kirsteins, I., "Data adaptive signal estimation by singular value decomposition of data matrix," *Proceedings of the IEEE*, vol. 70, no. 6, pp. 684–685, June 1982.

Tufts, D. W. and Shah, A. A., "Rank determination in time-series analysis," in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP)*, Apr. 1994, pp. IV–21–IV–24.

Wilkinson, J. H., *The Algebraic Eigenvalue Problem*. London, UK: Oxford Univ. Press, 1965.