Reduced Rank Filtering Techniques for Processing Multi-Aperture Radar

By

Peng Seng Tan

B.ENG (Honors), Electrical Engineering, National University of Singapore, 1995

M.Sc., Electrical and Electronic Engineering, Nanyang Technological University, 2001

Submitted to the Department of Electrical Engineering and Computer Science and the faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science

Thesis Committee

____

Dr. James M. Stiles (Chair)

____

Dr. Christopher Allen

____

Dr. Shannon Blunt

Date of Thesis Defense: 08/08/06

The Thesis Committee for Peng Seng Tan certifies

That this is the approved version of the following thesis

Reduced Rank Filtering Techniques for Processing Multi-Aperture Radar

Thesis Committee

____

Dr. James M. Stiles (Chair)

____

Dr. Christopher Allen

____

Dr. Shannon Blunt

Date Approved: __________

ACKNOWLEDGEMENTS

I will like to thank my MSc Thesis advisor, Prof Stiles for his invaluable guidance in the field of research work. Over the course of my research work, he has taught me on having the proper approach and mentality to tackle any research problem as well as the skills to present the findings in a coherent and unambiguous manner. Coming from a background of working as an engineer for nearly 10 years when the most cost effective solution is always the one and only goal, the acquiring of a research mentality is especially important for me since the goal of research is much wider than that of the goal of the final implementation. Besides that, Prof Stiles has also taught me several aspects of the radar and microwave system as well as SAR concepts either via formal coursework, discussions or book club readings. These measures have greatly enhanced my academic learning in the MSc degree that I am pursuing.

Next, I will also like to thank both Prof Allen and Prof Blunt who agree to be in my committee, especially Prof Allen after I have only approached him towards the end of my thesis preparation. At the same time, taking two radar related courses under Prof Allen has greatly improved my understanding in the field of radar system and signal processing as well and I am thankful for his enthusiasm and helpful advice shown in these classes. I will also like to thank Prof Blunt for offering his advice in some problems that I have faced during my research work even though he hardly knows me at all. At the same time, I have also benefited indirectly from his course notes in the advanced signal processing and these notes will be kept with me for a long time as well.

Thirdly, I will also like to thank my wife and children for supporting me throughout the course of my MSc degree. This is especially after I have spent so little time with them during these 2 years, being in KU in the day most of the times and missing many important family events that meant a lot to them. At the same time, coming to America with me has required my family to have very large adjustments to their daily lives which they have taken in their stride.

Fourthly, I will like to thank my fellow graduate students, Ambika Nanda and Vishal Sinha for their conversations and companionship, be it related to our research work or in the daily routines or cultures of our lives. They have certainly enhanced my overall experience here in America.

Finally, I will like to thank GOD who has given me the ability to acquire new skills and knowledge as well as enjoying the process of learning and the satisfaction of obtaining the end results. Without HIM, I will not even be in America to pursue this dream of higher education that I am currently involved in.

TABLE OF CONTENTS

TABLE OF FIGURES

ABSTRACT

Using a non-uniformly distributed aperture radar system for forming a SAR image will result in data correlations between the SAR image resolution cells. Thus, this requires that a more robust filter than the Matched Filter, i.e. the MMSE or Wiener Filter to be used in the receiver processing. As the Wiener Filter involves a computationally expensive matrix inverse operation, it can be avoided by using the Kalman filter. But the error covariance matrix computation in the Kalman filter can become unstable from finite machine precision in conjunction with large variations of the covariance matrix Eigen values. However, this instability can be overcome by using the Square Root Covariance Filter (SRCF) that ensures that the resulting error covariance matrix will always remain positive definite after each measurement update.

Besides the Kalman filter, a recent algorithm, namely the Multi-Stage Wiener Filter (MSWF) has been developed to overcome the matrix inverse problem in the Wiener Filter as well. Using orthogonal projections in each successive stage of decomposition, this filter is proven to achieve the same performance as the Wiener filter in a shorter computation period.

In this thesis, the performance of the SRCF and the MSWF used to form a SAR image is evaluated as compared to the Wiener filter and the Kalman filter using data from an existing radar model simulator. In addition, the use of reduced rank techniques is applied to both algorithms so as to trade off between computation time and accuracy. From the results obtained, the Reduced Rank Square Root Covariance Filter (RRSQRT) is able to achieve nearly the same performance as the Kalman filter in terms of accuracy even when the rank is reduced by 73%. However, as the Eigen decomposition needed for the reduced rank technique in the RRSQRT takes quite a while, thus the computation time for the RRSQRT is worse off than the Kalman filter if the time needed for Eigen decomposition is taken into account. At the same time, it is also shown that the MSWF is able to outperform the Wiener filter in terms of computation time while achieving the same accuracy level. Furthermore, this increase in computation performance is more prominent as the length of the measurement data increases. However, the computation time needed for the MSWF is still not able to match that of the Kalman filter although innovative implementation of the MSWF will help to narrow the gap between the 2 types of filter.

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

To obtain radar images from either ground based or airborne systems, there are clear advantages in replacing a large, monolithic, complex and single aperture radar system with much smaller but numerous distributed aperture radar sensors, (i.e., micro-sensors) in terms of cost, expansion capabilities as well as Mean Time Between Failures (MTBF) [1]. For example, let's consider the situation where it is required to obtain radar images from a space based sensor system. In terms of cost, building several small micro-satellites with lesser complex electronics as well as lower requirement of transmitter power per satellite will definitely be cheaper than building a large satellite that contains much complex electronics as well as high requirement of transmit power. In terms of expansion capabilities, it is always possible to increase the capability of the distributed aperture satellite radar system by adding more micro-satellites such that finer image resolution or larger imaging area can be easily achieved. In terms of failure rate, the malfunctioning of one single micro-satellite will not cripple the overall distributed aperture satellite radar system whereas any critical malfunctioning in the single aperture satellite radar system will render the complete failure of the system itself.

However, in terms of deployment, controlling the orbital trajectory of a single satellite is definitely much easier than controlling that of a constellation of micro-satellites. Thus, in general, it is not possible to have a constellation of micro-satellites flying in a regularly spaced orbit trajectory at all times as the orbital dynamics will make such a scenario near impossible. Instead, the actual deployment scenario will likely be a cluster of satellites flying in a non-uniform irregular spaced manner such that the positions of all the satellites will not form a regular spaced regular grid etc. When the distributed aperture radar system is operating in such a manner, this will greatly impact the Total Ambiguity Function (TAF) as the range-Doppler ambiguities will still be present in the Total Ambiguity Function of the system [1]. Refer to figure 1.1 and figure 1.2 below for the Total Ambiguity Function for a uniformly distributed aperture radar system versus that of a non-uniformly distributed aperture radar system. Note that both figures are obtained from [1] and reproduced in this document in order to illustrate the point mentioned above.

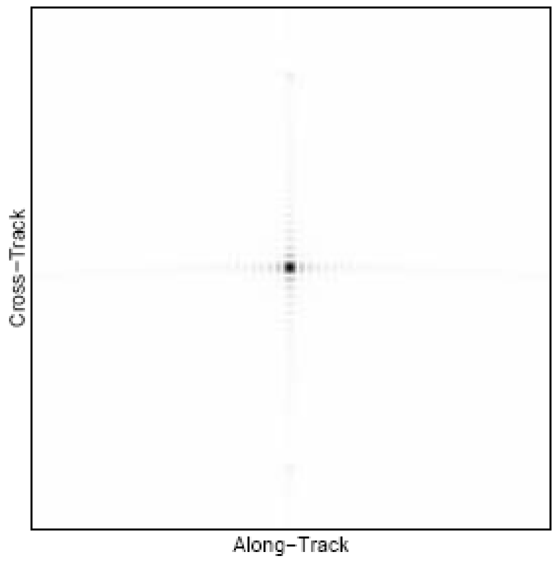


Figure 1-1: TAF for Regularly Spaced Distributed Apertures

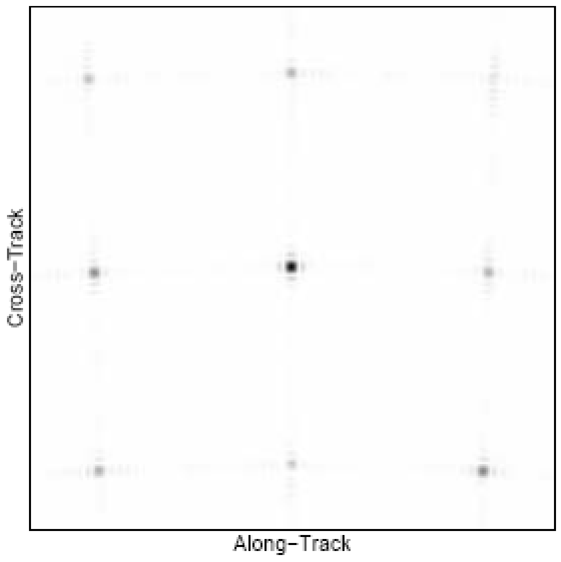


Figure 1-2: TAF for Non-Uniformly Spaced Distributed Apertures

Now, when the system's Total Ambiguity Function is as shown in figure 1.2, this will imply that each image resolution cell in a SAR image will be highly correlated to its neighboring resolution cells. As such, using a Matched Filter (MF) for the receiver channel processing will not produce a good quality image since the MF does not take data correlations between resolution cells into account. Thus, a more intelligent and robust filter will need to be used for the non-uniformly spaced distributed aperture radar system. From previous work in [1], it has been ascertained that the Minimum Mean Square Error filter (MMSE) or Wiener filter, is the appropriate filter choice.

Next, the MMSE filter is a much more complex filter than the standard Matched filter. Furthermore, the MMSE filter contains a matrix inverse operation on the correlation matrix of the measurement data. When the size of the measurement data is significant, i.e. in terms of thousands of measurements, the time needed for the matrix inverse operation will be computationally very expensive and may be impossible to execute for some hardware systems. Thus, we may ask whether it is possible to develop a more efficient MMSE filter for implementation. One answer to the above question is the Kalman filter that was developed during 1960. Essentially, the Kalman filter is like a recursive or iterative version of the MMSE filter in that the total measurement data set can be broken into many smaller blocks and the Kalman filter then performs the filter operation iteratively on each of the smaller block of data until all the available measurement data are utilized. In addition, when there are new incoming measurement data, the Kalman filter can refine the answers obtained from the previous measurement data using just this new information without having to rerun the whole set of measurement data again. As a result, the Kalman filter becomes very popular in many signal processing applications.

However, based on the findings in [2], [3] and various other literatures, it has been found that the results obtained from the Kalman filter can diverge or even the total failure of the Kalman recursion can occur as a result of finite computer precision. As a result of round-off errors due to finite precision, the error covariance matrix in the Kalman filter may end up having negative Eigen values although it is not possible for this condition to occur in theory. This ill condition is further magnified in instances when the measurements are very accurate or when there is a large range in magnitudes of the error covariance matrix's Eigen values. Thus, this motivates researchers to look into alternative recursive structures for propagating and updating the state estimates and the error covariance matrix. One of such alternative approach is to update the square root of the error covariance matrix at each iteration step instead of the error covariance matrix as in the Kalman filter. This approach, term the Square Root Covariance Filter (SRCF), is found to be able to yield two times as much effective precision as the Kalman filter in ill-conditioned problems [2]. Furthermore, the SRCF is also able to maintain the positive semi-definiteness of the error covariance matrix after each iteration step.

Nevertheless, the advantages of the square root covariance filter over the Kalman filter comes with the price of longer computation time for the same level of accuracy in the results obtained. Thus, this motivates researchers to examine the structure of the SRCF so as to further reduce its computational time such that it is comparable to that of the Kalman filter without any significant loss in the accuracy of the results obtained. One possibility is to compute a reduced rank of the square root of the error covariance matrix at each iteration step rather than the full rank at the expense of some slight loss in accuracy.

Besides utilizing the Kalman filter as a more efficient version of the MMSE filter for implementation, in recent years there are also other developments in the signal processing communities like the principal component algorithm or the cross-spectral algorithm that are Eigen-based methods. One of this development resulted in what is known as the Multi-Stage Wiener Filter (MSWF) that utilizes a decomposition for each stage that is based on orthogonal projections [4] [5]. Using the multi-stage Wiener filter, it is possible to avoid the matrix inverse operation present in the Wiener filter and this method also does not rely on any Eigen decomposition operation. At the same time, it is able to outperform other Eigen-based algorithms in terms of computational time and accuracy. An interesting feature of the multi-stage Wiener filter is that it has the reduced rank processing inherently built into its structure. For example, when the rank of the measurement data correlation matrix is equal to the value N, performing N stages of decomposition with the MSWF is equivalent to performing a full-rank operation of the filter. On the other hand, performing K stages of decomposition where $K < N$ is equivalent to performing a reduced rank operation of the filter up to rank K. Furthermore, it has also been determined in [5] that the number of decomposition stages required by the MSWF for achieving full rank MMSE performance is less than the full rank of the measured data correlation matrix. Thus, by controlling the number of the decomposition stages, a faster computational time can be obtained using the MSWF. Another feature of the MSWF is that it can be implemented in scalar format for predicting one variable at a time or in vector format for predicting groups of data variables simultaneously.

1.2 MOTIVATION OF THESIS

Coming back to the need of using the MMSE or Wiener filter for processing the received data from a Non-Uniformly Distributed Aperture Radar system, prior work on using the Kalman filter to replace the Wiener filter has been done by other researchers as shown in [6]. For my research area of interest, I will be looking into using the Reduced Rank Square Root Covariance filter (RRSQRT) in the receiver for SAR processing and compare its performance with the conventional Kalman filter. For this part of the work, both scalar and vector measurement update will be examined in each iteration. In addition, I will also be adopting the multi-stage Wiener filter used mostly by the communications community for SAR processing as well and compare its performance with both the traditional Wiener filter and the Kalman filter. For the multi-stage Wiener filter, both the scalar version and the vector version will also be looked at to compare their performance with the Kalman filter. Finally, in the process of implementing these algorithms, characteristics of each filter will be examined so as to allow the personality of each algorithm to be revealed.

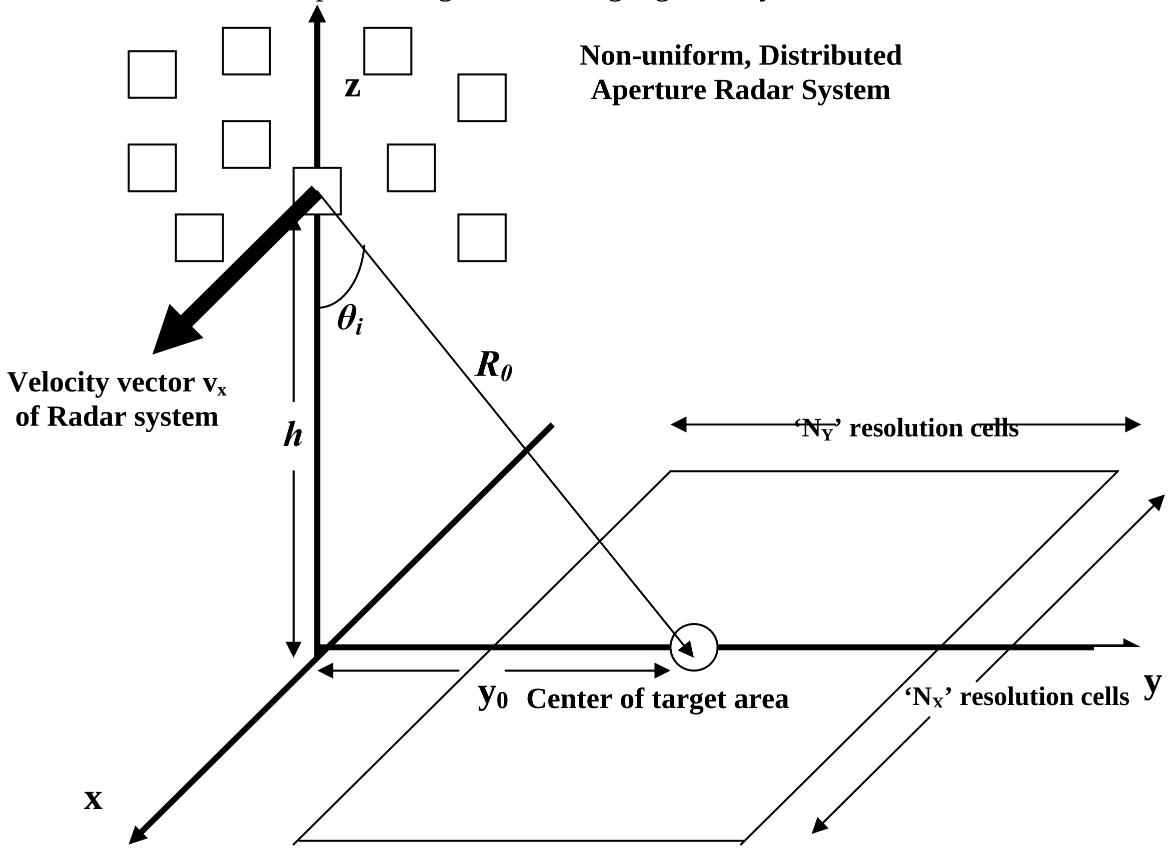
1.3 OUTLINE OF THESIS

In this thesis, Chapter 1 will cover the background and research motivation for the thesis itself. In Chapter 2, a brief description of the simulated radar model used for my research work will be provided. At the same time, the derivations of the equations that correspond to the Wiener filter and the Kalman filter implementations for the radar model will also be given as well. In Chapter 3, the equations pertaining to the square root covariance filter and its reduced rank adaptation will be provided and results obtained from using the SRCF and RRSQRT will be presented for discussions. Next, in Chapter 4, the equations for the multi-stage Wiener filter pertaining to my implementation will be shown and results obtained from using the scalar and vector implementations of the MSWF will be provided for comparison with that of the Wiener filter and Kalman filter. At the same time, a larger measurement data set is also simulated to further evaluate the performance of the multi-stage Wiener filter with the other 2 filters. In Chapter 5, some innovative pre-processing approaches for re-implementing the MSWF will be attempted and these preprocessing include either using a different method of initializing the parameters or re-ordering of the image resolution cell information based on correlation level before the actual stages of decomposition. Also, the structure "Recursive MSWF" will be introduced as one of the innovative approach within the Chapter itself. Finally, Chapter 6 will conclude on the findings of this thesis along with further recommendations.

CHAPTER 2: THE RADAR MODEL

2.1 DISTRIBUTED APERTURE AND TARGET GEOMETRY

Now, due to a lack of real data from a non-uniformly distributed radar system, a radar model simulator was created using MATLAB by prior research students that allows the users to have the flexibility to model each aspect of the radar system independently. For the distributed apertures within the radar system used in my research, it is modeled as a single transmitter and 12-receivers radar system. The figure 2.1 below provides an illustration of the geometry of the transmitter and receivers location in space along with the target geometry.



z
Non-uniform, Distributed Aperture Radar System
θ_i
R_0
Velocity vector v_x of Radar system
h
'N_Y' resolution cells
y_0 Center of target area
'N_X' resolution cells
y
x

Figure 2-1: Distributed Aperture Radar System and Target Geometry

Now, in this geometry, certain parameters are defined as follows:

- The overall distributed aperture radar configuration is moving in the *x* direction with a velocity vector v_x.
- The center of the radar configuration is at the location, $x = 0$, $y = 0$ and $z = h$. Thus, the radar configuration is flying at a height *h* above the ground.
- The center of the target area of interest is at the location, $x = 0$, $y = y_0$ and $z = 0$. Also, flat earth geometry is assumed such that *z* is equal to zero throughout the entire target area.
- The target area is modeled as a square grid in where there are N_x and N_y ($N_x = N_y$) square image resolution cells in each dimension for a total of $N_t = N_x \times N_y$ image resolution cells. Note that the area of each image resolution cell is $\Delta x \times \Delta y$ where Δx is the along track resolution and Δy is the cross track resolution.
- The angle θ_i is the incident angle to the center of the target area of interest. As such, we can define the parameters y_0 and *h* as:

$$y_0 = R_0 \sin\theta_i \tag{2.1}$$

$$h = R_0 \cos\theta_i \tag{2.2}$$

Besides these parameters, the locations of both the transmitter and receivers are obtained from a Gaussian random generator in MATLAB with zero mean and variances set to a value that will ensure that the spatial extent of all the distributed apertures will not be so large as to affect the image resolution. In addition, care is taken to ensure that each receiver aperture is not too close to the others so as to allow each receiver aperture to collect independent measurement samples.

2.2 SIGNAL SPACE MODELING

2.2.1 TRANSMIT SIGNAL MODELING

Details of the overall signal space modeling can be found in [8]. Below is a brief description of the different signal parameters used in the model itself that can also be obtained from [9]. Firstly, the transmit signal function $\vec{s}$ will be modeled as a vector that has a total number of elements equal to $J \times BT$ where J is the number of transmitter, B is the transmit bandwidth and T is the total duration of the transmit pulses used for forming the image. It can be expressed as a summation of basis functions as follows:

$$\vec{s}(\overline{x}_t) = \sum_{n=1}^{N} s_n \, \hat{\phi}_n(\overline{x}_t) \qquad (2.3)$$

In equation 2.3, N is the total number of transmit signals and equal to $J \times BT$ and the vector $\overline{x}_t$ is a 5-D vector for the transmitter representing the combination of spatial coordinates x, y and z as well as the slow time t and fast frequency w variables:

$$\overline{x} = [t,w,x,y,z] \qquad (2.4)$$

Next, the basis functions $\hat{\phi}_n$ are functions of space, time and frequency that span the total time width, bandwidth and transmit aperture size as a whole. As for the weight parameters s_n, they can be considered as complex coefficients for the basis functions that will determine the explicit form of the total transmit signal $\vec{s}$. As such the transmit signal function $\vec{s}$ can be represented in vector form as:

$$\mathbf{s} = [s_1,s_2,s_3,\ldots\ldots,s_N]^{\mathrm{T}} \qquad (2.5)$$

2.2.2 TARGET MODELING

Now, the target area of interest consists of N_t number of image resolution cells. Thus, the target function can also be modeled as a summation of N_t basis functions as:

$$\vec{\gamma}(\overline{x}_s) = \sum_{m=1}^{N_t} \gamma_m \, \vec{\psi}_m(\overline{x}_s) \qquad (2.6)$$

In equation 2.6, the individual basis function $\vec{\psi}_m$ represents the point scatterer in each image location of the total image area. Also, the complex variable γ_m represents the scattering coefficient associated with each image resolution cell location. Thus, in vector format, the target function is expressed as:

$$\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \gamma_3, \gamma_4, \ldots, \gamma_{Nt}]^T \qquad (2.7)$$

2.2.3 RESPONSE MEASUREMENTS MODELING

For the measurements obtained at the different receivers, they can be generally expressed as a space-time function using the convolution integral:

$$\vec{r}(\overline{x}_r) = \int \overleftrightarrow{H}(\overline{x}_r;\overline{x}_s).\vec{\gamma}(\overline{x}_s).\int \overleftrightarrow{G}(\overline{x}_s;\overline{x}_t).\vec{s}(\overline{x}_t)d\overline{x}_t d\overline{x}_s + \vec{n}(\overline{x}_r) \qquad (2.8)$$

In the above equation, the functions $\overleftrightarrow{H}(\overline{x}_r;\overline{x}_s)$ and $\overleftrightarrow{G}(\overline{x}_s;\overline{x}_t)$ are the dyadic Green's functions that describe the propagation from radar to target area and back to the receivers. Also, the function $\vec{n}(\overline{x}_r)$ represents the Gaussian receiver noise present in the system. Now, similarly the function $\vec{r}(\overline{x}_r)$ can also be expressed as a summation of basis functions:

$$\vec{r}(\overline{x}_r) = \sum_{l=1}^{M} r_l \, \vec{\varphi}_l(\overline{x}_r) \tag{2.9}$$

where *M* is the total number of measurements from all the receivers and the values r_l are complex and equal to:

$$r_l = \int \vec{r}(\overline{x}_r).\vec{\varphi}_l(\overline{x}_r) d\overline{x}_r \tag{2.10}$$

Using a series of manipulations, it is possible to obtain the final format of the response measurements in the following format:

$$\begin{aligned} \mathbf{r} &= \sum_{t}^{N_t} \gamma_t \mathbf{H}_t \mathbf{s} + \mathbf{n} \\ &= \sum_{t}^{N_t} \gamma_t \boldsymbol{\rho}_t + \mathbf{n} \end{aligned} \tag{2.11}$$

In the above equation, **n** is the receiver Gaussian noise, $\mathbf{H}_t$ is a matrix and $\boldsymbol{\rho}_t = \mathbf{H}_t\mathbf{s}$ is the normalized space-time receiver measurement from the *t*-th scatterer or image resolution cell and these parameters are defined as follows:

$$\mathbf{r} = [r_1, r_2, r_3, r_4, \ldots.. r_M]^T \tag{2.12}$$

$$\mathbf{n} = [n_1, n_2, n_3, n_4, \ldots.. n_M]^T \tag{2.13}$$

$$\mathbf{H}_t = \begin{pmatrix} H^t{}_{11} & \ldots & H^t{}_{1N} \\ \vdots & \ddots & \vdots \\ H^t{}_{M1} & \cdots & H^t{}_{MN} \end{pmatrix} \tag{2.14}$$

$$H^t_{mn} = \int \vec{\varphi}_m(\overline{x}_t).\int \overleftrightarrow{H}(\overline{x}_r;\overline{x}_s).\vec{\psi}_t(\overline{x}_s).\int \overleftrightarrow{G}(\overline{x}_s;\overline{x}_t).\vec{\phi}_n(\overline{x}_t) d\overline{x}_t d\overline{x}_s d\overline{x}_r \tag{2.15}$$

By looking at equation 2.11, we can combine the normalized space-time receiver measurements or response vectors from all the scatterers or image resolution cells into a matrix **P**:

$$\mathbf{P} = [\rho_1;\rho_2;\rho_3;\rho_4;\ldots..;\rho_{Nt}] \tag{2.16}$$

Using equation 2.16, we can rewrite equation 2.11 as follows:

$$\mathbf{r} = \mathbf{P}\gamma + \mathbf{n} \tag{2.17}$$

2.3 THE SAR IMAGE FORMULATION

2.3.1 WIENER FILTER IMPLEMENTATION

Now, from the definitions obtained in the previous section, forming a SAR image from the response vectors **r** is simply to obtain the values $\hat{\gamma}$ (predicted scattering coefficients) from **r** using the linear filtering model:

$$\hat{\gamma} = \mathbf{W}\mathbf{r} \tag{2.18}$$

$$\hat{\gamma} = [\,\hat{\gamma}_{1,}\ \hat{\gamma}_{2,}\hat{\gamma}_{3,}\hat{\gamma}_{4,}\ldots..\hat{\gamma}_{N_t}\,] \tag{2.19}$$

$$\mathbf{W} = [\,\mathbf{w}_{\mathit{1}}\ \mathbf{w}_{\mathit{2}}\ \mathbf{w}_{\mathit{3}}\ldots\ldots\mathbf{w}_{N_t}\,]^H \tag{2.20}$$

In equation 2.18, **W** is the weight matrix/linear filtering operation that will extract the predicted values $\hat{\gamma}$ from the response vector and $\boldsymbol{w}_i$ is the weight vector corresponding to the ith image resolution cell or scatterer. Thus, to obtain the values of $\hat{\gamma}$ using a MMSE filter, we will need to define the weight matrix $\mathbf{W}_{\text{mmse}}$ in equation 2.20 that is linked to our signal model. As a start, we know from [10] that the weight vector for a MMSE filter is defined as:

$$\begin{aligned} \mathbf{w_i} &= \mathbf{R}^{-1}\mathbf{p}_i \\ \mathbf{R} &= \mathrm{E}[\mathbf{r}\mathbf{r}^{\mathbf{H}}] \\ \mathbf{p}_i &= \mathrm{E}[\mathbf{r}\gamma_i] \end{aligned} \qquad (2.21)$$

Where **R** is the measurement data correlation matrix and $\mathbf{p}_i$ is the cross correlation vector between the response vector **r** and the scattering coefficient γ_i of ith target. Also, it is assumed that the scattering coefficients and the receiver noise are statistically independent. Thus, using the same definition, we arrive at the expression for $\mathbf{W}_{\text{mmse}}$ as

$$\mathbf{W}_{\mathrm{mmse}} = \mathrm{E}[\gamma\gamma^{\mathrm{H}}]\mathbf{P}^{\mathrm{H}}(\mathbf{P}\mathrm{E}[\gamma\gamma^{\mathrm{H}}]\mathbf{P}^{\mathrm{H}} + \mathrm{E}[\mathbf{n}\mathbf{n}^{\mathrm{H}}])^{-1} \qquad (2.22)$$

If we assume that the elements of the vector $\boldsymbol{\gamma}$ are independent with identical statistical properties and E{$\mathbf{nn}^{\mathrm{H}}$} is equal to the noise covariance matrix $\mathbf{K}_{\mathrm{n}}$ since its mean value is zero, then equation 2.22 can be rewritten as:

$$\mathbf{W}_{\mathrm{mmse}} = \gamma_t^2\mathbf{P}^{\mathrm{H}}(\gamma_t^2\mathbf{P}\mathbf{P}^{\mathrm{H}} + \mathbf{K}_n)^{-1} \qquad (2.23)$$

in which $\gamma_t^2 = \mathrm{E}[\gamma_i^H\gamma_i]$ is the expected value of the square of the scattering coefficient of each image resolution cell or scatterer. Looking at the above equation 2.23, it can be seen that there is a matrix inverse operation associated with the MMSE filter and the size of the matrix is equal to $M\times M$ where M is the total number of measurements. Thus, for situations in which the amount of measurement data is significant, the calculation of the $\mathbf{W}_{\text{mmse}}$ will be computationally inefficient. Thus, a much more efficient method of implementing the MMSE filter of filter order O(2×M^3) will be to adapt the problem into a Kalman filter implementation.

2.3.2 KALMAN FILTER IMPLEMENTATION

As mentioned in the previous sections, Kalman filter is the iterative form of the MMSE or Wiener filter. In order to adapt the SAR image processing using the Kalman filter implementation, we will need to address a few issues. Firstly, the full segment of the response vector given in equation 2.12 will be divided into many smaller segments such that each smaller segment of the measurement data will be fed into one Kalman iteration until all the data are completely utilized. Similarly, we will need to do likewise for the noise vector defined in equation 2.13 as well as for the **P** matrix defined in equation 2.16. Assuming a total of L segments are to be obtained with 4 elements per segment as an example, the resulting partition of these 3 parameters will be as shown below:

$$\begin{aligned}
\mathbf{r} &= \left[\underbrace{r_1\ r_2\ r_3\ r_4}_{\mathbf{r}(1)},\underbrace{r_5\ r_6\ r_7\ r_8}_{\mathbf{r}(2)},\underbrace{r_9\ r_{10}\ r_{11}\ r_{12}}_{\mathbf{r}(3)},\ldots\ldots\ldots,\underbrace{r_{M-3}\ r_{M-2}\ r_{M-1}\ r_M}_{\mathbf{r}(L)}\right] \\
\mathbf{P} &= \left[\mathbf{P}(1)\ \mathbf{P}(2)\ \mathbf{P}(3)\ldots\ldots\ldots\ \mathbf{P}(L)\right]^T \qquad (2.24) \\
\mathbf{n} &= \left[\underbrace{n_1\ n_2\ n_3\ n_4}_{\mathbf{n}(1)},\underbrace{n_5\ n_6\ n_7\ n_8}_{\mathbf{n}(2)},\underbrace{n_9\ n_{10}\ n_{11}\ n_{12}}_{\mathbf{n}(3)},\ldots\ldots\ldots,\underbrace{n_{M-3}\ n_{M-2}\ n_{M-1}\ n_M}_{\mathbf{n}(L)}\right]
\end{aligned}$$

Secondly, it is known that there are two equations that define the Kalman filter implementation, namely the state equation and the observation equation. For the state equation used in our modeling, it is given by the equation in the following page:

$$\boldsymbol{\gamma}(l) = \mathbf{A}(l)\boldsymbol{\gamma}(l-1) + \mathbf{u}(l) \tag{2.25}$$

where **A**(*l*) is the state transition matrix and **u**(*l*) is the process noise that represents the uncertainty in **A**(*l*). Note that in this equation, *l* represents the iteration or data segment number out of a total of L iterations or segments. As for the observation equation, it can be defined as:

$$\mathbf{r}(l) = \mathbf{P}(l)\boldsymbol{\gamma}(l) + \mathbf{n}(l) \tag{2.26}$$

Thirdly, for our radar model, we are also assuming that the elements of the vector $\boldsymbol{\gamma}$ remain constant with respect to time, frequency and space over the total duration T of the measurements. As such, we can replace the state transition matrix **A**(*l*) with an identity matrix **I**. Besides that, other assumptions or initial conditions that are used for my Kalman filter implementation are as follows:

$$\begin{aligned}
&\hat{\gamma}(0/0) = \mathrm{E}\{\gamma\} = 0 \\
&\mathbf{K}_{\gamma}(0/0) = \sigma_{\gamma}^{2}\mathbf{I} \\
&\mathbf{K}_{n}(0/0) = \sigma_{n}^{2}\mathbf{I} \\
&\mathbf{K}_{\mathbf{u}}(l) = \mathrm{E}\{\mathbf{u}(l)\mathbf{u}(l)^{\mathrm{H}}\} = 0
\end{aligned} \tag{2.27}$$

These assumptions are based on the fact that firstly, we do not have any prior information about the values of the scattering coefficients and thus we can set them to be equal to zeros. Secondly, each scattering coefficient is independent of the other scattering coefficients. Thirdly, since each scattering coefficient is assumed to remain unchanged during the observation period, the model noise is thus assumed to be equal to zero as well. Fourthly, the measurement noise is assumed to be white Gaussian noise.

With the above definitions and assumptions made, it is now possible to define the Kalman filter implementation steps for our radar model. With the value of *l* set to 1 and using the initial conditions, the iteration steps begin as follows:

- Step 1: Updating the A priori Error covariance matrix

$$\mathbf{K}_{\gamma}(l/l-1) = \mathbf{K}_{\gamma}(l-1/l-1) + \mathbf{K}_{\mathbf{u}}(l) \qquad (2.28)$$

- Step 2: Computing the Kalman gain **G**(*l*)

$$\mathbf{G}(l) = \mathbf{K}_{\gamma}(l/l-1)\mathbf{P}(l)^{\mathrm{H}}[\mathbf{P}(l)\mathbf{K}_{\gamma}(l/l-1)\mathbf{P}(l)^{\mathrm{H}} + \mathbf{K}_{\mathbf{n}}(l)]^{-1} \qquad (2.29)$$

- Step 3: Obtaining the Innovation **v**(l) from the new measurement data **r**(*l*)

$$\mathbf{v}(l) = \mathbf{r}(l) - \mathbf{P}(l)\hat{\gamma}(l-1/l-1) \qquad (2.30)$$

- Step 4: Computing the updated predicted scattering coefficient $\hat{\gamma}(l/l)$ using the Kalman gain and innovation obtained from the previous steps

$$\hat{\gamma}(l/l) = \hat{\gamma}(l-1/l-1) + \mathbf{G}(l)\mathbf{v}(l) \qquad (2.31)$$

- Step 5: Computing the Error covariance matrix $\mathbf{K}_{\gamma}(l/l)$ for the current iteration

$$\mathbf{K}_{\gamma}(l/l) = [\mathbf{I} - \mathbf{G}(l)\mathbf{P}(l)]\mathbf{K}_{\gamma}(l/l-1) \qquad (2.32)$$

With the five steps defined above, the Kalman filter implementation for the SAR image processing of the non-uniformly distributed aperture radar system can be achieved.

2.4 TESTING THE KALMAN FILTER IMPLEMENTATION

2.4.1 1ST TEST SCENARIO FOR KALMAN FILTER

In order to ensure that the Kalman filter implementation described in the previous section is functionally properly for the SAR image processing, two test scenarios are carried out. In the 1st test scenario, an image consisting of random complex scattering coefficients γ and is generated and used as input to the radar model simulator to form the output measurement vector r. Other additional parameter values used for the two test scenarios are described in Table 2.1 below:

Table 2.1: Parameter Values used for Test Scenario 1 and 2

	Description of Parameters	**Values chosen**
1	h (Height of Distributed Aperture Radar System)	183 km
2	v_x (Velocity of Distributed Aperture Radar System)	7.8 km/s
3	f_c (Center Transmit frequency)	10 GHz
4	N_x	31
5	N_y	31
6	B (Bandwidth of Transmit signal)	0.3125 MHz
7	T (Time width of Transmit signal)	0.7327 ms
8	θ_i (Incident angle)	45°
9	R_o	258.8 km
10	J (Number of transmitters)	1
11	I (Number of receivers)	12
12	Number of samples per receiver	255

	Description of Parameters	**Values chosen**
13	Total number of samples for 12 receivers	3060
14	SNR (Signal to Noise Ratio)	40 dB
15	σ_{γ}^{2} for random complex scattering coefficients γ	2

Using the Kalman filter, the estimated scattering coefficient $\hat{\gamma}$ is obtained. To measure the performance of the Kalman filter, a numerical parameter known as the Normalized Mean Square Error or MSE is then computed at each step of the iteration:

$$MSE = \frac{(\hat{\gamma} - \gamma)^{\mathrm{H}}(\hat{\gamma} - \gamma)}{\hat{\gamma}^{\mathrm{H}}\gamma} \tag{2.33}$$

Next, the result of the computed MSE from the Test scenario 1 is as shown in figure 2.2 along with the expected error from the Kalman filter obtained from normalizing the trace of the error covariance matrix with the same denominator as in equation 2.33, i.e.

$$MSE\_\mathrm{cov} = \frac{\mathrm{trace}(\mathbf{K}_{\gamma}(l))}{\hat{\gamma}^{\mathrm{H}}\gamma} \tag{2.34}$$

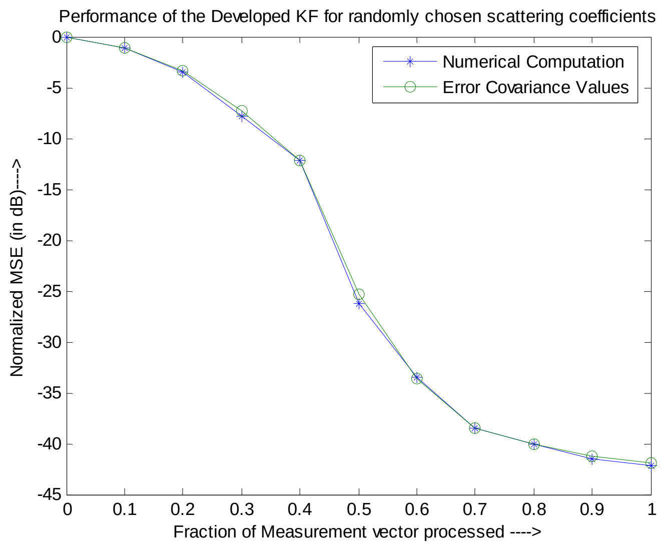


Figure 2-2: Performance of KF for randomly chosen scattering coefficients

Note that a functional check on the validity of the Kalman filter results is the degree of correlation between the two parameters, MSE and MSE_cov defined in equation 2.33 and 2.34. From figure 2.2, it can be seen that these 2 parameters are closely tracking one another and thus it is concluded that the results from the Kalman filter for the 1st test scenario is valid.

2.4.2 2nd TEST SCENARIO FOR KALMAN FILTER

Now, in the 2nd test scenario for the Kalman filter implementation, a small section of a photograph of the KU football stadium was cropped to a size of 31×31 and each image resolution cell's intensity is given a random phase such that the resulting set of the 31×31 complex image resolution cell values is now used as the γ input to the radar model simulator to form the output measurement vector r with σ_γ^2

equal to 128^2 in this scenario. Figure 2.3 below gives a view of these new input scattering coefficients γ used for the 2nd test.

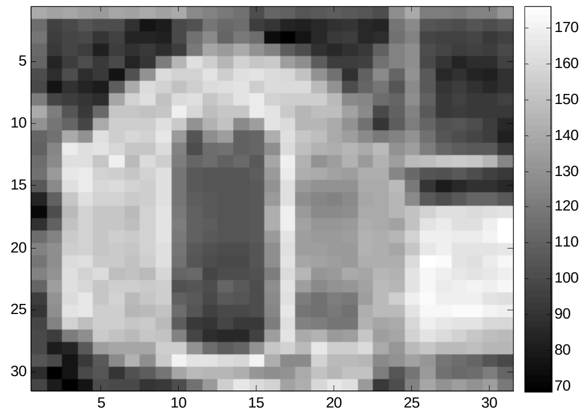


Figure 2-3: Actual Image used to generate γ and Response Vectors r

With all other factors remaining constant, a Kalman filter run is executed on this new set of data and the results are shown in figure 2.4 and figure 2.5 on the next page.

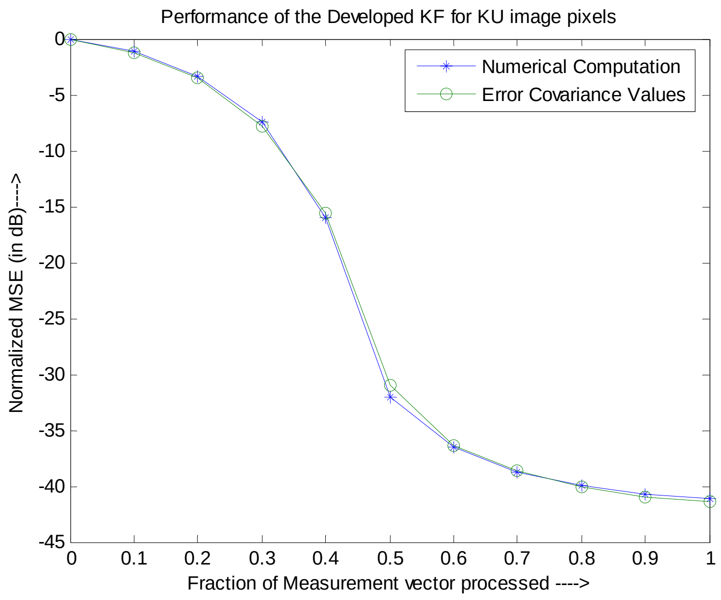


Figure 2-4: Performance of KF for KU image resolution cells used as Input

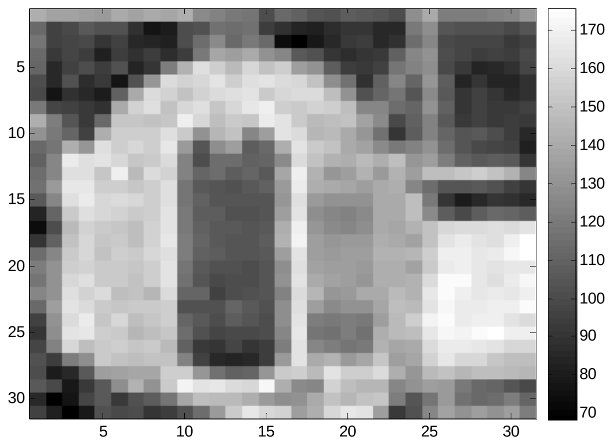


Figure 2-5: Estimated image obtained from KF

By examining the two curves in figure 2.4, it can also be concluded that the Kalman filter is working properly for the 2nd test scenario and this is further verified by the similarity of the output image obtained from the Kalman filter shown in figure 2.5 to the input image shown in figure 2.3. Thus, the two functional tests performed on the Kalman filter implementation of the SAR image formation using simulated model data based on non-uniformly distributed aperture radar system are successful.

2.5 LOOKING BEYOND THE CONVENTIONAL KALMAN FILTER

As mentioned in Chapter 1, although the Kalman filter is also to produce results at a much faster rate as compared to the Wiener filter, it may also suffer from some shortcomings that will cause it to diverge or fail completely. These pitfalls are well recorded in [2] and [11]. In order to circumvent the problems faced by the Kalman filter, researchers have derived new forms of recursive filters from the original structure of the Kalman filter and one new form of recursive filter is the Square Root Covariance Filter (SRCF). For this filter, besides updating the state estimate at each iteration step, the square root of the error covariance matrix is updated instead of the error covariance matrix itself. From experiments, researchers have proven that the SRCF is showing better numerical precision and stability as compared to the Kalman filter, especially in ill-conditioned problems. At the same time, the computational time and memory needed by the SRCF is not significantly greater than that of the Kalman filter. Thus, this new form of recursive filter is desirable as an alternative to the Kalman filter for the SAR image formation application for the non-uniformly distributed aperture radar system.

In the next Chapter, I will describe the equations that define the SRCF developed by Potter for scalar measurement update per iteration as well as its alternate form. I will then proceed to define the vector measurement update version of Potter's SRCF that was developed by Andrew. As prior work done in [6] has demonstrated that it is computationally faster to implement the Kalman filter using vector measurement update per iteration as compared to scalar measurement update, thus I will use the results obtained using both Potter and Andrew's version of filter to compare with the scalar and vector measurement update versions of the Kalman filter.

Next, to address the issue of the larger computation time required by the SRCF as compared to the Kalman filter, I will also explore using a reduced rank version of the SRCF that was proposed in [12] and compare the results obtained from different adaptations of the reduced rank version with that of the full rank version.

CHAPTER 3: THE SQUARE ROOT COVARIANCE FILTER

3.1 FULL RANK SQUARE ROOT FILTER

In Chapter 2, it is shown that the conventional Kalman filter can be utilized for the SAR image formation of the non-uniformly distributed aperture radar system. However, it is also mentioned that there are some pitfalls with the conventional Kalman filter in terms of stability and divergence issues. As such, researchers have developed different forms of iterative filters to overcome the pitfalls of the Kalman filter. One of these filters is the Square Root Covariance filter (SRCF) developed by Potter [2] that was initially used for space navigation purposes in which the process noise in the state equation is equal to zero. At the same time, Potter also confined the operations of his SRCF to only scalar measurement updates. Essentially in the SRCF, the error covariance matrix $\mathbf{K}_\gamma(l-1/l-1)$ and $\mathbf{K}_\gamma(l/l-1)$ can be represented in the following manner:

$$\begin{aligned}\mathbf{K}_\gamma(l-1/l-1) &= \mathbf{S}(l-1/l-1)\mathbf{S}(l-1/l-1)^{\mathrm{H}} \\ \mathbf{K}_\gamma(l/l-1) &= \mathbf{S}(l/l-1)\mathbf{S}(l/l-1)^{\mathrm{H}}\end{aligned} \qquad (3.1)$$

In which the matrices $\mathbf{S}(l-1/l-1)$ and $\mathbf{S}(l/l-1)$ are the respective square roots of the Error Covariance Matrices. Using the above definitions, therefore, it is possible to rewrite equation 2.28 as the following:

$$\mathbf{S}(l/l-1)\mathbf{S}(l/l-1)^{\mathrm{H}} = \mathbf{S}(l-1/l-1)\mathbf{S}(l-1/l-1)^{\mathrm{H}} \qquad (3.2)$$

$$\mathbf{S}(l/l-1) = \mathbf{S}(l-1/l-1) \qquad (3.3)$$

since the process noise covariance matrix $\mathbf{K}_{\mathbf{u}}(l)$ is set to zero in our radar model. From equation 3.2, we can see that the appropriate propagation variables for the SRCF will be $\mathbf{S}(l/l-1), \mathbf{S}(l/l)$ in addition to the predicted scattering coefficients $\hat{\gamma}(l/l)$ and $\hat{\gamma}(l-1/l-1)$. Also, the update of the error covariance matrix $\mathbf{K}_{\gamma}(l/l)$ defined in equation 2.32 can be rewritten as:

$$\mathbf{S}(l/l)\mathbf{S}(l/l)^{\mathrm{H}} = \mathbf{S}(l/l-1)[\mathbf{I} - b(l)\mathbf{a}(l)\mathbf{a}(l)^{\mathrm{H}}]\mathbf{S}(l/l-1)^{\mathrm{H}} \quad (3.4)$$

Where the scalar variable *b*(*l*) and the vector **a**(*l*) are defined by:

$$\mathbf{a}(l) = \mathbf{S}(l/l-1)^{\mathrm{H}}\mathbf{p}(l)^{\mathrm{H}} \quad (3.5)$$

$$1/b(l) = \mathbf{a}(l)^{\mathrm{H}}\mathbf{a}(l) + \sigma_n^2(l) \quad (3.6)$$

Furthermore, Potter has shown that it is possible to factor the bracket term $[\mathbf{I} - b(l)\mathbf{a}(l)\mathbf{a}(l)^{\mathrm{H}}]$ into a product of 2 terms such that:

$$\begin{aligned} [\mathbf{I} - b\mathbf{a}\mathbf{a}^{\mathrm{H}}] &= [\mathbf{I} - b\eta\mathbf{a}\mathbf{a}^{\mathrm{H}}][\mathbf{I} - b\eta\mathbf{a}\mathbf{a}^{\mathrm{H}}]^{\mathrm{H}} \\ \eta &= 1\Big/\left(1 + \sqrt{b\sigma_n^2}\right) \end{aligned} \quad (3.7)$$

Note that in equation 3.7, η is a scalar variable. By using the factorization, the equation 3.4 can then be simplified to equation 3.8 shown below:

$$\begin{aligned} \mathbf{S}(l/l) &= \mathbf{S}(l/l-1)[\mathbf{I} - b(l)\eta(l)\mathbf{a}(l)\mathbf{a}(l)^{\mathrm{H}}] \\ &= \mathbf{S}(l/l-1) - b(l)\eta(l)\mathbf{S}(l/l-1)\mathbf{a}(l)\mathbf{a}(l)^{\mathrm{H}} \end{aligned} \quad (3.8)$$

By combining equations 3.5 to 3.8, it is now possible to define the square root covariance filter implementation for our radar model. With the value of *l* set to 1 and using the initial conditions as per the Kalman filter implementation, the iteration steps can begin as follows:

$$
\begin{aligned}
\mathbf{a}(l) &= \mathbf{S}(l\,/\,l-1)^{\mathrm{H}}\,\mathbf{p}(l)^{\mathrm{H}} \\
b(l) &= 1\big/[\mathbf{a}(l)^{\mathrm{H}}\,\mathbf{a}(l) + \sigma^{2}{}_{\mathbf{n}}(l)] \\
\eta(l) &= 1\big/[1 + \{b(l)\sigma^{2}{}_{\mathbf{n}}(l)\}^{1/2}] \\
\mathbf{g}(l) &= b(l)\mathbf{S}(l\,/\,l-1)\mathbf{a}(l) \\
\hat{\gamma}(l\,/\,l) &= \hat{\gamma}(l-1\,/\,l-1) + \mathbf{g}(l)[r(l) - \mathbf{p}(l)\hat{\gamma}(l-1\,/\,l-1)] \\
&= \hat{\gamma}(l-1\,/\,l-1) + \mathbf{g}(l)v(l) \\
\mathbf{S}(l\,/\,l) &= \mathbf{S}(l\,/\,l-1) - \eta(l)\mathbf{g}(l)\mathbf{a}(l)^{\mathrm{H}}
\end{aligned}
\tag{3.9}
$$

In the above iteration process, **g**(*l*) is the Kalman gain vector and *v*(*l*) is the scalar innovation as per the conventional Kalman filter. In this instance, the initial value of $\mathbf{S}(0\,/\,0)$ is obtained by taking the matrix square root of $\mathbf{K}_{\gamma}(0\,/\,0)$. Before proceeding further, it is noted in [2] that an alternate structure of equation 3.9 is often employed as well and this structure is as shown in equation 3.10 below:

$$
\begin{aligned}
\mathbf{a}(l) &= \mathbf{S}(l\,/\,l-1)^{\mathrm{H}}\,\mathbf{p}(l)^{\mathrm{H}} \\
\sigma(l) &= [\mathbf{a}(l)^{\mathrm{H}}\,\mathbf{a}(l) + \sigma^{2}{}_{\mathbf{n}}(l)]^{1/2} \\
\alpha(l) &= \sigma(l) + \sqrt{\sigma^{2}{}_{\mathbf{n}}(l)} \\
\beta(l) &= 1\big/(\sigma(l)\alpha(l)) \\
\mathbf{g}(l) &= \beta(l)\mathbf{S}(l\,/\,l-1)\mathbf{a}(l) \\
\hat{\gamma}(l\,/\,l) &= \hat{\gamma}(l-1\,/\,l-1) + \mathbf{g}(l)[\alpha(l)\big/\sigma(l)][r(l) - \mathbf{p}(l)\hat{\gamma}(l-1\,/\,l-1)] \\
&= \hat{\gamma}(l-1\,/\,l-1) + \mathbf{g}(l)[\alpha(l)\big/\sigma(l)]v(l) \\
\mathbf{S}(l\,/\,l) &= \mathbf{S}(l\,/\,l-1) - \mathbf{g}(l)\mathbf{a}(l)^{\mathrm{H}}
\end{aligned}
\tag{3.10}
$$

Mathematically, it is not difficult to prove that the iteration steps shown in equation 3.10 are equivalent to that of equation 3.9. Furthermore, equation 3.10 can readily be converted to a vector measurement update structure that the generic Potter's SRCF does not address in equation 3.9.

Next, from previous research done on the performance of the conventional Kalman filter [6], it has been observed that the performance of the Kalman filter in terms of computational time will improve significantly when the iteration steps are performed using vector measurement updates rather than scalar measurement updates. However, once the size of the measurement update block exceeds a certain threshold, the savings achieved in the computational time for the vector measurement update implementation will decrease gradually. Eventually, it will become the MMSE filter where all measurements are used in 1 iteration step or update. To illustrate this point, a small simulation is carried out with the Kalman filter using a total of 504 measurements (response vector $\mathbf{r}$ is 504×1) to predict an image size of 255 (15×15) resolution cells. Next, a parameter L is defined where L is equal to the number of smaller vectors the response vector $\mathbf{r}$ is divided into. By increasing the value of L from a value of 1 corresponding to the MMSE filter until it is equal to 504 corresponding to the total amount of measurements, the time taken to obtain the final result for each case is recorded. These results are then plotted and shown in figure 3.1 on the following page.

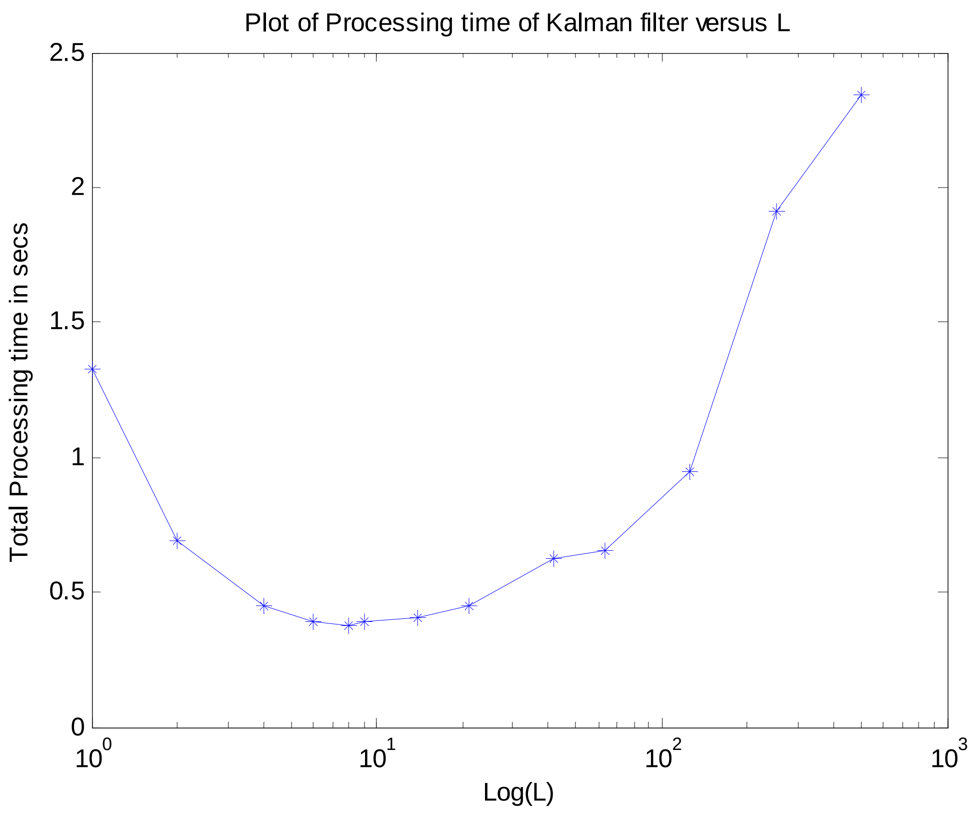


Figure 3-1: Processing Time of Kalman filter versus *L*

By examining figure 3.1, it can be seen that the Kalman filter is least efficient when it is implemented using scalar measurement updates. The main reason is because the total amount of iterations required for the scalar measurement updates is significant although the time required for each iteration step is small. Thus, this blows up the total processing time as seen from the figure where the time taken is about 2.35 sec. On the other hand, when all the measurements are used in just one iteration step (L = 1) as seen at the leftmost point in the figure 3.1, the time required for the matrix inverse operation will predominate.

Therefore, based on the observation shown in figure 3.1 above as well as in [6], it will be logical to implement the Kalman filter using vector measurement updates of reasonable block sizes rather than using scalar measurement updates. Since in my research, I am exploring using the SRCF to replace the Kalman filter, thus it also creates a need to search for the vector measurement update version of the Potter's SRCF. The search has resulted in finding the Andrew's SRCF [2] that can process either scalar or vector measurement updates in each iteration step. The equations for Andrew's SRCF implementation are as shown below:

$$\begin{aligned}
&\mathbf{A}(l) = \mathbf{S}(l\,/\,l-1)^{\mathrm{H}}\,\mathbf{P}(l)^{\mathrm{H}} \\
&\boldsymbol{\Sigma}(l) = \sqrt{\mathbf{A}(l)^{\mathrm{H}}\,\mathbf{A}(l) + \mathbf{K}_{\mathbf{n}}(l)}^{\,c} \\
&\hat{\gamma}(l\,/\,l) = \hat{\gamma}(l-1\,/\,l-1) + \mathbf{S}(l\,/\,l-1)\mathbf{A}(l)[\boldsymbol{\Sigma}(l)^{-1}]^{\mathrm{H}}\,\boldsymbol{\Sigma}(l)^{-1}\,\mathbf{v}(l) \\
&\mathbf{S}(l\,/\,l) = \mathbf{S}(l\,/\,l-1) - \mathbf{S}(l\,/\,l-1)\mathbf{A}(l)[\boldsymbol{\Sigma}(l)^{-1}]^{\mathrm{H}}[\boldsymbol{\Sigma}(l) + \sqrt{\mathbf{K}_{\mathbf{n}}(l)}]\mathbf{A}(l)^{\mathrm{H}}
\end{aligned} \tag{3.11}$$

By examining the equations in 3.11, we can see its similarity to that of the alternate form of Potter's implementation shown in equation 3.10. Note that the matrix $\boldsymbol{\Sigma}(l)$ is obtained by taking the lower triangular Cholesky decomposition of the expression $(\mathbf{A}(l)^{\mathrm{H}}\,\mathbf{A}(l) + \mathbf{K}_{\mathbf{n}}(l))$.

3.2 PERFORMANCE BETWEEN KALMAN AND FULL RANK SRCF

In the previous section, the equations for both the scalar measurement update and vector measurement update implementations of the Square Root Covariance filter are derived. In this section, I will show the results of a simulation being carried out for these filters using their full rank and compared their performance with that of the Kalman filter using both the scalar and vector measurement update implementations. For this simulation, some of the test parameters are as shown in Table 3.1 below. Also, the image used is composed of random complex scattering coefficients γ.

Table 3.1: Parameter Values used for SRCF Simulation

	Description of Parameters	**Values chosen**
1	N_x	31
2	N_y	31
3	Full Filter Rank size (= $N_x \times N_y$)	961
4	Total number of transmitters	1
5	Total number of receivers	12
6	Total number of samples	3060
7	SNR (Signal to Noise Ratio)	40 dB

In this simulation, as the total number of available measurements is 3060, thus some efforts are spent to obtain a reasonable measurement update size to be used for the vector update implementation as is discussed in the previous section. The final values that are decided are 102 measurements per update for a total of 30 iterations. Next, the types of filters that are being put to the test are as shown in Table 3.2 in the following page:

Table 3.2: Types of Filter used for SRCF Simulation

	Description of Filter	**Type of Measurement Update**	**Filter Order**
1	Kalman filter Type 1	Scalar	$O(N_t^2)$
2	Potter SRCF	Scalar	$O(N_t^2)$
3	Andrew SRCF Type 1	Scalar	$O(N_t^2)$
4	Kalman filter Type 2	Vector, 102 measurements per block	$O(N_t^2 \times M/L)$
5	Andrew SRCF Type 2	Vector, 102 measurements per block	$O(N_t^2 \times M/L)$

For the SRCF simulation, the Potter SRCF and Andrew SRCF Type 1 are executed to verify the results obtained from the SRCF with the Kalman filter Type 1. For performance measure in terms of computational time and accuracy of results, the Andrew SRCF Type 2 is used to compare with that obtained from the Kalman filter Type 2. The results that are obtained from this simulation are as shown below:

Table 3.3: Results obtained for SRCF Simulation

	Description of Filter	**Time/sec**	**Final MSE/dB**
1	Kalman filter Type 1	356.350	-42.216
2	Potter SRCF	437.451	-42.216
3	Andrew SRCF Type 1	488.223	-42.216
4	Kalman filter Type 2	37.206	-42.216
5	Andrew SRCF Type 2	44.320	-42.216

Besides the tabulation of results in Table 3.3, figure 3.2 to figure 3.5 also shows the plots of the MSE and MSE_cov defined in equation 2.33 and 2.34 versus the amount of data processed for the Potter SRCF, Andrew SRCF Type 1, Kalman filter Type 2 and Andrew SRCF Type 2.

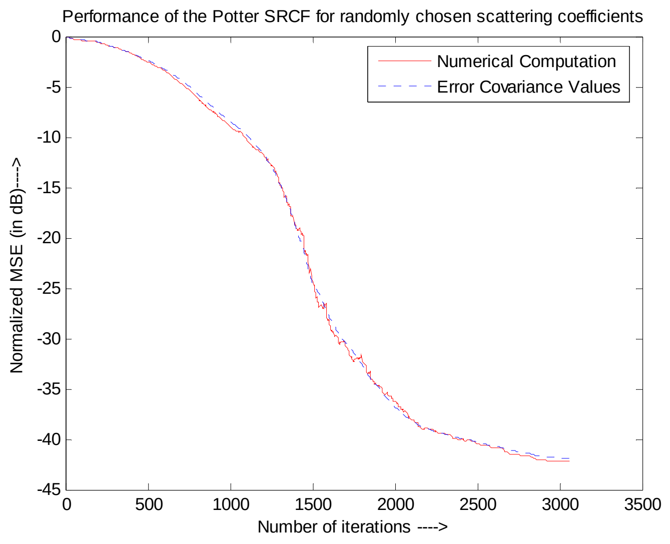


Figure 3-2: Performance of the Potter SRCF

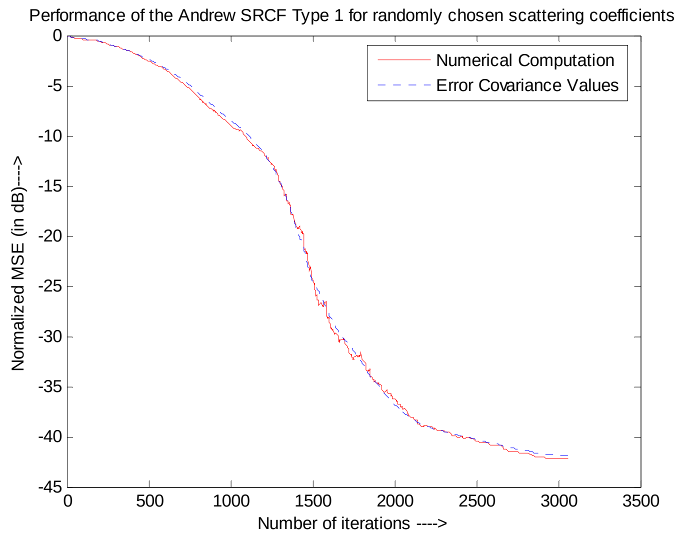


Figure 3-3: Performance of the Andrew SRCF Type 1

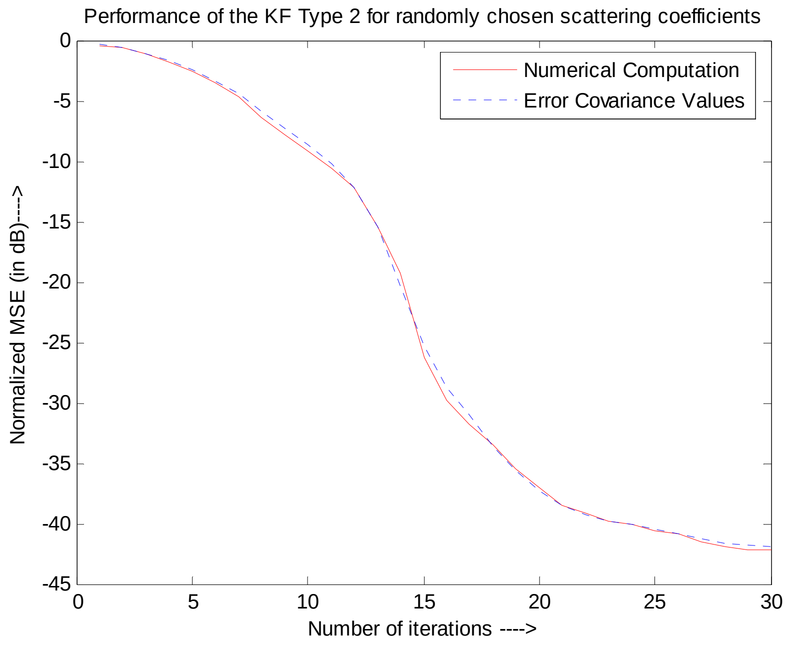


Figure 3-4: Performance of the Kalman filter Type 2

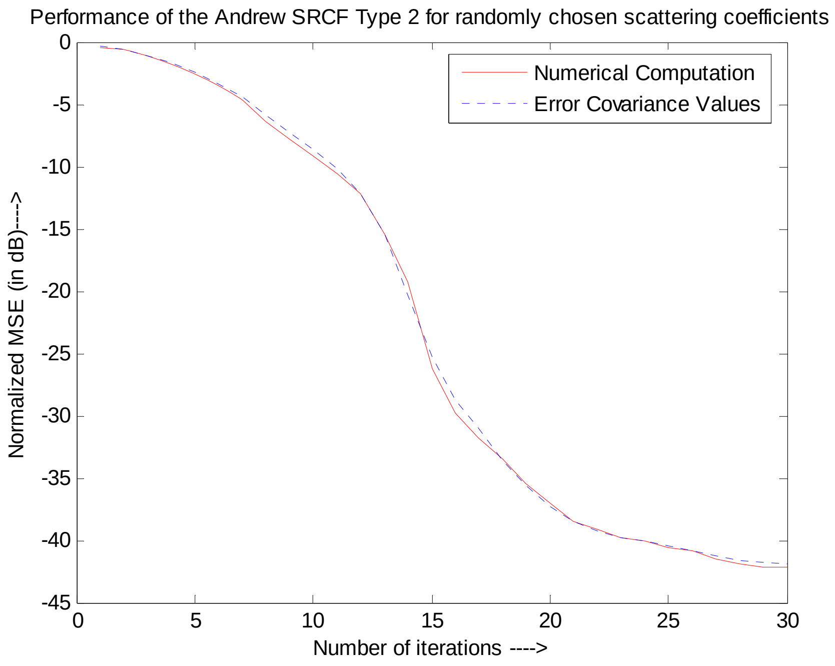


Figure 3-5: Performance of the Andrew SRCF Type 2

By comparing the results obtained from the Potter SRCF, Andrew SRCF Type 1 with that of the conventional Kalman filter Type 1 as shown in Table 3.3, it can be seen that in terms of accuracy of results obtained, all 3 filters are equivalent in terms of performance. However, in terms of computational time, the conventional Kalman filter will outperform the other two filters as is mentioned in Chapter 2 and [2] etc. Also, by examining the time obtained using the scalar measurement update implementation with that of vector measurement update implementation, one can readily see the huge improvement in computational time using the later implementation.

Next, by comparing the results obtained between the Kalman filter Type 2 with that of the Andrew SRCF Type 2 (both using vector measurement updates), we can see that the performance of both filters are equivalent in terms of accuracy of results but the Kalman filter Type 2 is superior in terms of computational time. Thus, after looking at all the results above, it can be concluded that the SRCF can be used to replace the Kalman filter for SAR image formation using a non-uniformly distributed aperture radar system. Although the SRCF incurs a higher computational time than the Kalman filter, it offers better numerical stability and thus the trade off between the 2 performance parameters is acceptable.

At this point in time, the next research question that comes into my mind will be to look into approaches to reduce the computational time of the SRCF such that it will be comparable to the Kalman filter. One possibility is that the improvement in computational efficiently may be achieved by a slight degradation of the accuracy of results. In the next section, this approach will be examined in details.

3.3 REDUCED RANK SQUARE ROOT FILTER

As mentioned in the previous section, there is a motivation to look into new approaches to the SRCF that will reduce its computational time while maintaining its accuracy of results as much as possible. After some intensive research, an answer is found in the papers [12] to [14] by M.Verlaan and A.W. Heemink. In their papers, they have presented the technique of using a reduced rank approximation of the square root of the error covariance matrix to compute the final results with little degradation in the accuracy of the results. To achieve the process of rank reduction of the square root covariance matrix, it involves operations for determining the q dominant Eigen values and Eigen vectors of the square root covariance matrix. Once these q dominant Eigen values and vectors are obtained, they are then used to form the new square root covariance matrix whereas the (n-q) less dominant Eigen values and vectors of the square root covariance matrix are discarded with n being the total number of Eigen vectors/values or the full rank of the matrix. In this manner, rank reduction is achieved since only the q dominant Eigen vectors/values are used to form the square root covariance matrix. At the same time, lesser computation time is required to compute the square root covariance matrix since the matrix column dimension of the square root covariance matrix have also been reduced from n to q.

Next, the reduced rank technique or Reduced Rank Square Root filter (RRSQRT) as it is called by M. Verlaan consists of 3 main steps that are as follows:

1. Time-step
2. Reduction-step
3. Measurement-step

For a better visualization of each of this step that together forms a complete RRSQRT iteration step, they will be explained in more details in the following sections.

3.3.1 TIME-STEP OF RRSQRT

In the Time-step portion, the update of the approximate a priori square root error covariance matrix $\tilde{\mathbf{S}}(l\,/\,l-1)$ that is equivalent to equation 3.3 is performed:

$$\tilde{\mathbf{S}}(l\,/\,l-1) = [\tilde{\mathbf{S}}(l-1\,/\,l-1), \mathbf{K}_{\mathrm{u}}(l)^{1/2}] \qquad (3.12)$$

In our radar model, since we have set the process noise covariance matrix $\mathbf{K}_{\mathbf{u}}(l)$ to be equal to zero, thus equation 3.12 will be similar to equation 3.3. The main difference lies in the fact that the square root error covariance matrix $\tilde{\mathbf{S}}(l-1\,/\,l-1)$ in equation 3.12 has only q_{l-1} columns instead of n columns due to a Reduction-step process in the previous iteration step. Thus, it is already an approximation of the full rank $\mathbf{S}(l-1\,/\,l-1)$. Also, compared to the full rank SRCF, the number of computations needed for obtaining the approximate square root covariance matrix has been reduced by a factor of $\dfrac{n}{q_{l-1}}$.

3.3.2 REDUCTION-STEP OF RRSQRT

In the Reduction-step portion, an Eigen decomposition is performed on the product of $\tilde{\mathbf{S}}(l/l-1)^{\mathrm{H}}\tilde{\mathbf{S}}(l/l-1)$ so as to obtain the Eigen vectors and Eigen values of the approximate Error covariance matrix $\tilde{\mathbf{K}}_{\gamma}(l/l-1)$. The operation is expressed in equation 3.13 where $\tilde{\mathbf{U}}(l)$ and $\tilde{\mathbf{D}}(l)$ are the Eigen vectors and values of $\tilde{\mathbf{S}}(l/l-1)^{\mathrm{H}}\tilde{\mathbf{S}}(l/l-1)$:

$$\tilde{\mathbf{S}}(l/l-1)^{\mathrm{H}}\tilde{\mathbf{S}}(l/l-1) = \tilde{\mathbf{U}}(l)\tilde{\mathbf{D}}(l)\tilde{\mathbf{U}}(l)^{\mathrm{H}} \qquad (3.13)$$

Although $\tilde{\mathbf{K}}_{\gamma}(l/l-1)$ is equal to $\tilde{\mathbf{S}}(l/l-1)\tilde{\mathbf{S}}(l/l-1)^{\mathrm{H}}$ and the Eigen decomposition should be performed on this expression such that $\tilde{\mathbf{K}}_{\gamma}(l/l-1)$ is given by $\tilde{\mathbf{V}}(l)\tilde{\mathbf{D}}(l)\tilde{\mathbf{V}}(l)^{\mathrm{H}}$ where $\tilde{\mathbf{V}}(l)$ are the Eigen vectors of $\tilde{\mathbf{K}}_{\gamma}(l/l-1)$, it has been shown in [15] that the two expressions $\tilde{\mathbf{S}}(l/l-1)^{\mathrm{H}}\tilde{\mathbf{S}}(l/l-1)$ and $\tilde{\mathbf{S}}(l/l-1)\tilde{\mathbf{S}}(l/l-1)^{\mathrm{H}}$ has the same nonzero Eigen values. Also, the Eigen vectors $\tilde{\mathbf{V}}(l)$ of $\tilde{\mathbf{K}}_{\gamma}(l/l-1)$ are given by the expression [$\tilde{\mathbf{S}}(l/l-1)\tilde{\mathbf{U}}(l)\tilde{\mathbf{D}}(l)^{-1/2}$]. As it is faster to compute the Eigen decomposition of $\tilde{\mathbf{S}}(l/l-1)^{\mathrm{H}}\tilde{\mathbf{S}}(l/l-1)$ that has only a matrix size of $(q_{l-1} \times q_{l-1})$ rather than $(n \times n)$, thus the above operation in equation 3.13 is performed. Next, the reduction process of $\tilde{\mathbf{S}}(l/l-1)$ is performed by only retaining the dominant q_l Eigen vectors in the matrix $\tilde{\mathbf{U}}(l)$ based on certain threshold or criteria where $q_l \leq q_{l-1}$ such that:

$$\tilde{\mathbf{S}}_{\mathbf{c}}(l/l-1) = [\tilde{\mathbf{S}}(l/l-1)\tilde{\mathbf{U}}(l)]_{1:n,1:q_l} \tag{3.14}$$

Now, for equation 3.14, the expression $[]_{1:n,1:q}$ means the truncation of columns inside the matrix [] to q_l number of columns.

3.3.3 MEASUREMENT-STEP OF RRSQRT

Next, in the measurement-step portion of the RRSQT, it is similar to the iteration steps of the Potter SRCF or the Andrew SRCF except that the expression $\mathbf{S}(l/l-1)$ in the full rank SRCF is replaced by the expression $\tilde{\mathbf{S}}_{\mathbf{c}}(l/l-1)$ for the RRSQRT. The RRSQRT version for the Andrew SRCF Type 2 is as shown below:

$$\begin{aligned}
&\mathbf{A}(l) = \tilde{\mathbf{S}}_{\mathbf{c}}(l/l-1)^{\mathrm{H}}\mathbf{P}(l)^{\mathrm{H}} \\
&\mathbf{\Sigma}(l) = \sqrt{\mathbf{A}(l)^{\mathrm{H}}\mathbf{A}(l) + \mathbf{K}_{\mathbf{n}}(l)}^{c} \\
&\hat{\gamma}(l/l) = \hat{\gamma}(l-1/l-1) + \tilde{\mathbf{S}}_{\mathbf{c}}(l/l-1)\mathbf{A}(l)[\mathbf{\Sigma}(l)^{-1}]^{\mathrm{H}}\mathbf{\Sigma}(l)^{-1}\mathbf{v}(l) \\
&\tilde{\mathbf{S}}(l/l) = \tilde{\mathbf{S}}_{\mathbf{c}}(l/l-1) - \tilde{\mathbf{S}}_{\mathbf{c}}(l/l-1)\mathbf{A}(l)[\mathbf{\Sigma}(l)^{-1}]^{\mathrm{H}}[\mathbf{\Sigma}(l) + \sqrt{\mathbf{K}_{\mathbf{n}}(l)}]\mathbf{A}(l)^{\mathrm{H}}
\end{aligned} \tag{3.15}$$

Finally, for initialization purposes, the value of $\tilde{\mathbf{S}}(0/0)$ can be obtained by using the q leading Eigen vectors and values of the Error covariance matrix $\mathbf{K}_{\gamma}(0/0)$ so that

$$\begin{aligned}
&\mathbf{K}_{\gamma}(0/0) = \mathbf{V}(0)\mathbf{D}(0)\mathbf{V}(0)^{\mathrm{H}} \\
&\tilde{\mathbf{S}}(0/0) = [\mathbf{V}(0)\mathbf{D}(0)^{1/2}]_{1:n,1:q}
\end{aligned} \tag{3.16}$$

At the same time, it is also stated in [14] that when the value of q is set to be equal to the value of n in each iteration step, the RRSQRT of filter order $O(N_t \times q_{l-1}^2 + 4\times(M/L)^3)$ will revert to the full rank SRCF that is also mathematically equivalent to the conventional Kalman filter.

3.4 CRITERIA DETERMINATION FOR RRSQRT

In section 3.3, the equations pertaining to the RRSQRT are discussed in details. However, one very important aspect that is needed to implement the RRSQRT with respect to our radar model has not yet been decided. Essentially, this aspect is the choosing of the criteria for determining the number of dominant Eigen vectors and values to be retained after each iteration step. In order to choose this criteria for the RRSQRT, two approaches are used in the investigation process. The 1st approach is basically just using guesswork to determine the number of q dominant Eigen vectors and values in each of the RRSQRT iteration step right from the initialization process. The 2nd approach is much more systematic and involves extracting the Eigen spectrum of the Square Root Error covariance matrix using the full rank SRCF before deciding on the most appropriate criteria to be used. To evaluate the performance of the RRSQRT using these two approaches, the simulation setups and results of each approach are presented in section 3.4.1 and 3.4.2

3.4.1 USING GUESSWORK FOR CRITERIA DETERMINATION

In this section, the 1st approach of using guesswork to determine the criteria for retaining the q dominant Eigen vectors and values is discussed. As it has been verified in section 3.2 that the SRCF of full rank n is equivalent in accuracy with the Kalman filter, some possibilities for choosing q can be just by setting it to be $n \times 0.5$ or any number less than n right at the initialization process stated in equation 3.16. Thus, for the 1st approach, the parameter q is set to the following values of 1, 40 and

$n \times 0.5$ where n is equal to 961 for a 31×31 image resolution cell SAR scenario at initialization and no further reduction step is carried out throughout the whole iteration process. It is my belief that these 3 values will provide a good picture for understanding the behavior of the RRSQRT pertaining to the radar model. At the same time, for completeness of the investigation process, the 1st approach is applied to the Potter SRCF Type 1, Andrew SRCF Type 1 that are scalar measurement update based and Andrew SRCF Type 2 that is based on vector measurement update. Below are the results obtained in the simulations using these 3 values of q for each of the 3 types of filter with the same initial conditions as in Table 3.1 under section 3.2:

Table 3.4: Results obtained for RRSQRT Simulation using 1st approach

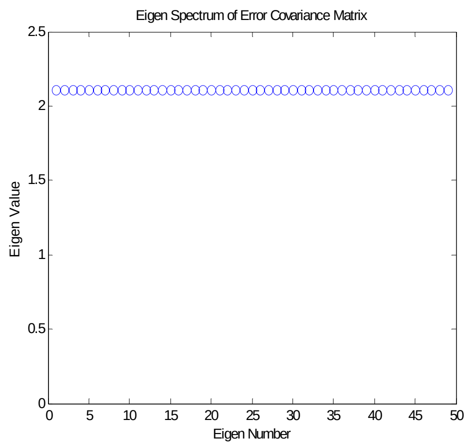
	Filter Type	**Value of *q***	**Time/sec**	**Final MSE/dB**
1	Potter SRCF Type 1	1	5.322	0.0027
2	Andrew SRCF Type 1	1	5.531	0.0027
3	Andrew SRCF Type 2	1	6.397	0.0027
4	Potter SRCF Type 1	40	24.563	-0.0986
5	Andrew SRCF Type 1	40	23.762	-0.0986
6	Andrew SRCF Type 2	40	8.078	-0.0986
7	Potter SRCF Type 1	480	225.381	-1.9
8	Andrew SRCF Type 1	480	253.819	-1.9

	Filter Type	**Value of *q***	**Time/sec**	**Final MSE/dB**
9	Andrew SRCF Type 2	480	25.035	-1.9

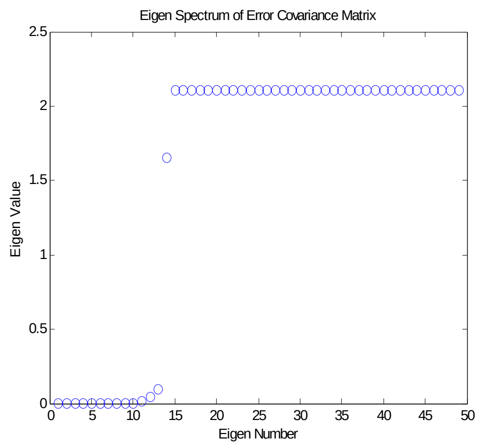
From the results obtained in Table 3.4 compared to that in Table 3.3 for full rank SRCF, one can conclude that the dominant number of Eigen vectors and values of $\mathbf{K}_{\gamma}(0/0)$ is definitely greater than half of its matrix rank. This conclusion is based on the bad results achieved for the final MSE even for test case 7, 8 and 9. Thus, using the guesswork approach does not provide a good solution for determining the criteria and thus the 2nd approach is then explored.

3.4.2 USING SRCF EIGEN SPECTRUM FOR CRITERIA DETERMINATION

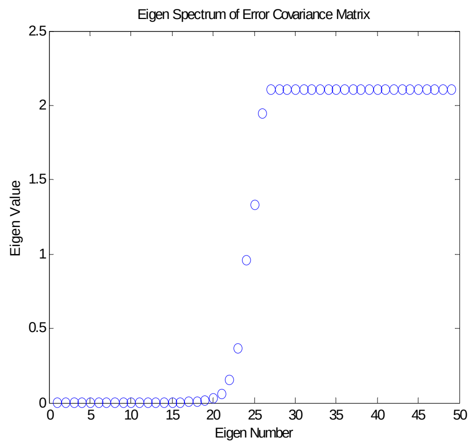
In order to determine the behavior of the Eigen values of the SRCF as it steps through each iteration, a simulation is carried out using an input image consisting of 7*7 image resolution cells along with 120 measurements. This will mean that the size of the Error covariance matrix $\mathbf{K}_{\gamma}(0/0)$ is 49 rows by 49 columns. The rationale for choosing this image size is that there are already 49 Eigen values contained in $\mathbf{K}_{\gamma}(0/0)$ and thus this amount of Eigen values is sufficient for my investigation. By performing an Eigen decomposition at various stages of the whole iteration process, the Eigen Spectrum of the Error covariance matrix is obtained with respect to the % of total iteration process. The results obtained are then plotted out for analysis of the variation of the dominant Eigen values as the iteration progresses.



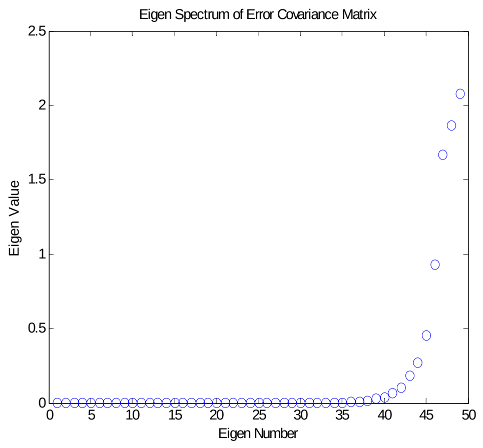
0 % of iteration process



12.5% of iteration process



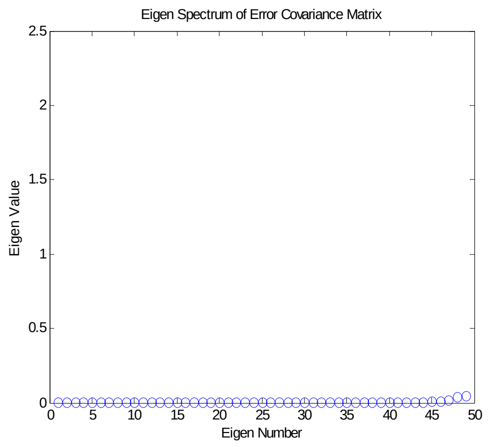
25% of iteration process



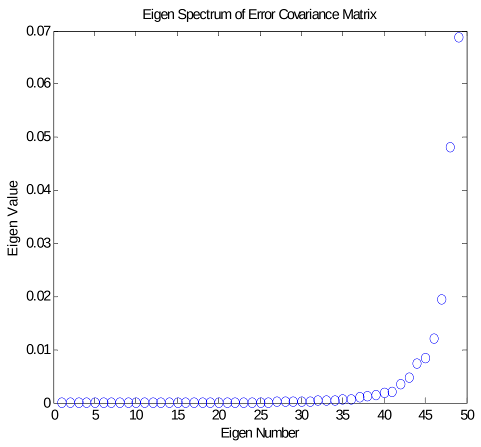
50% of iteration process

Figure 3-6: Eigen Spectrum at 0% to 50% of Total iteration process

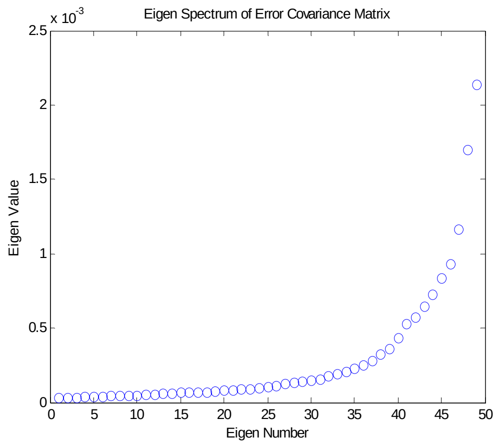
By looking at the Eigen Spectrum shown in Figure 3.6 at the beginning of the iteration, we can see that all 49 Eigen values are the same initially and thus all the corresponding Eigen vectors are dominant. However, as the iteration progresses along, some of the Eigen values decrease in magnitude until they become insignificant. Thus, after some point in time, the dominant number of Eigen vectors will decrease below the value of 49.



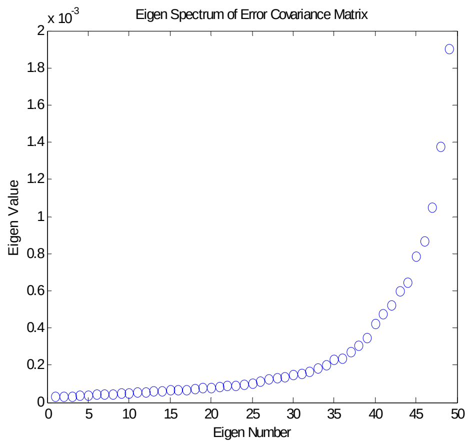
62.5% of iteration process



62.5% of iteration process (Zoom in)



90% of iteration process (Zoom in)



100% of iteration process (Zoom in)

Figure 3-7: Eigen Spectrum at 62.5% to 100% of Total iteration process

Next, by looking at Figure 3.7, we can see that once the iteration process reaches 62.5% of the data, there are hardly any dominant Eigen vectors left in the Error covariance matrix and thus it can be approximated by a rank 1 matrix. By observing the behavior of the Eigen Spectrum as the iteration progresses, it has put me in a good position to develop the criteria for determining the *q* number of dominant Eigen vectors to be retained in each iteration step. Essentially, it should not be based on a constant number *q* but should rely on some threshold levels. At the same time, as it is observed that the Eigen Spectrum will not change significantly within 1 iteration step, thus there is no need to perform an Eigen decomposition process and reduction-step process at every iteration step. Instead, both processes can be performed based on certain step-size changes in the MSE etc, i.e. applying another criterion for performing these 2 processes. After much considerations, both the criterions for performing the Eigen decomposition process and reduction-step process (criteria 1) and for determining the value of *q* (criteria 2) are derived based on a systematic and engineering approach to the problem. For criteria 1, it will be that the 2 processes are applied at every 4 dB changes in the expected value of the MSE. For criteria 2, it will be based on the number of Eigen values exceeding a threshold based on a certain % in magnitude of the initial Eigen value. The details of both criterions are combined and shown in Table 3.5 below:

Table 3.5: Details of Criterion for 2nd approach

% of initial Eigen Value used for Criteria 2	**Step size in dB for MSE variation used for Criteria 1**				
	4	8	12	16	20
0.001					
0.01					

% of initial Eigen Value used for Criteria 2	**Step size in dB for MSE variation used for Criteria 1**				
	4	8	12	16	20
0.05					
0.1					
0.5			EXAMPLE CASE		
1.0					
4.0					

From Table 3.5, it can be seen that the combination of the two criterions will result in a total of 35 test scenarios. Thus, it is believed that these test scenarios will provide a comprehensive picture of the performance of the RRSQRT in the SAR image formation problem. Also, before proceeding to execute the test scenarios for the RRSQRT, one point to note is that for criteria 2, at least one Eigen vector corresponding to the largest Eigen value of the Square Root Error covariance matrix should be retained at all times after applying the criteria to discard the less dominant Eigen vectors. Otherwise, with an empty $\tilde{\mathbf{S}}_{\mathbf{c}}(l/l-1)$ matrix, it will be impossible to complete the remaining measurement-step process of the current iteration step as well as the subsequent iteration steps.

3.5 PERFORMANCE OF REDUCED RANK SQUARE ROOT FILTER

In the previous sections, all the equations pertaining to the RRSQRT as well as the criterions needed for the implementation have been derived. Also, it has been determined that there will be a total of 35 test scenarios to be performed on the RRSQRT so as to evaluate its performance in terms of computational time and accuracy of final results obtained. With all these preparation, a simulation is then performed that covers all the test scenarios. In the simulation, 3 parameters are then tracked for performance measure and they are the numerical MSE, the total computational time as well as the number of remaining Eigen vectors at the end of the iteration process for each test scenario. The results recorded are then shown in Table 3.6 to Table 3.8. At the same time, a 3-dimensional plot of the results shown in each Table is also displayed in Figure 3.8 to 3.10.

Table 3.6: Results of MSE (dB) obtained for RRSQRT simulation

% of initial Eigen Value used for Criteria 2	**Step size in dB for MSE variation used for Criteria 1**				
	4	8	12	16	20
0.001	-42.216	-42.216	-42.216	-42.216	-42.216
0.01	-39.906	-40.649	-40.818	-41.210	-41.144
0.05	-32.726	-35.422	-36.855	-38.222	-38.148
0.1	-27.014	-31.374	-34.708	-36.600	-36.742
0.5	-14.399	-21.629	-27.134	-31.176	-32.489

% of initial Eigen Value used for Criteria 2	**Step size in dB for MSE variation used for Criteria 1**				
	4	8	12	16	20
1.0	-11.039	-17.247	-25.312	-28.999	-30.699
4.0	-2.239	-13.008	-17.616	-26.687	-27.348

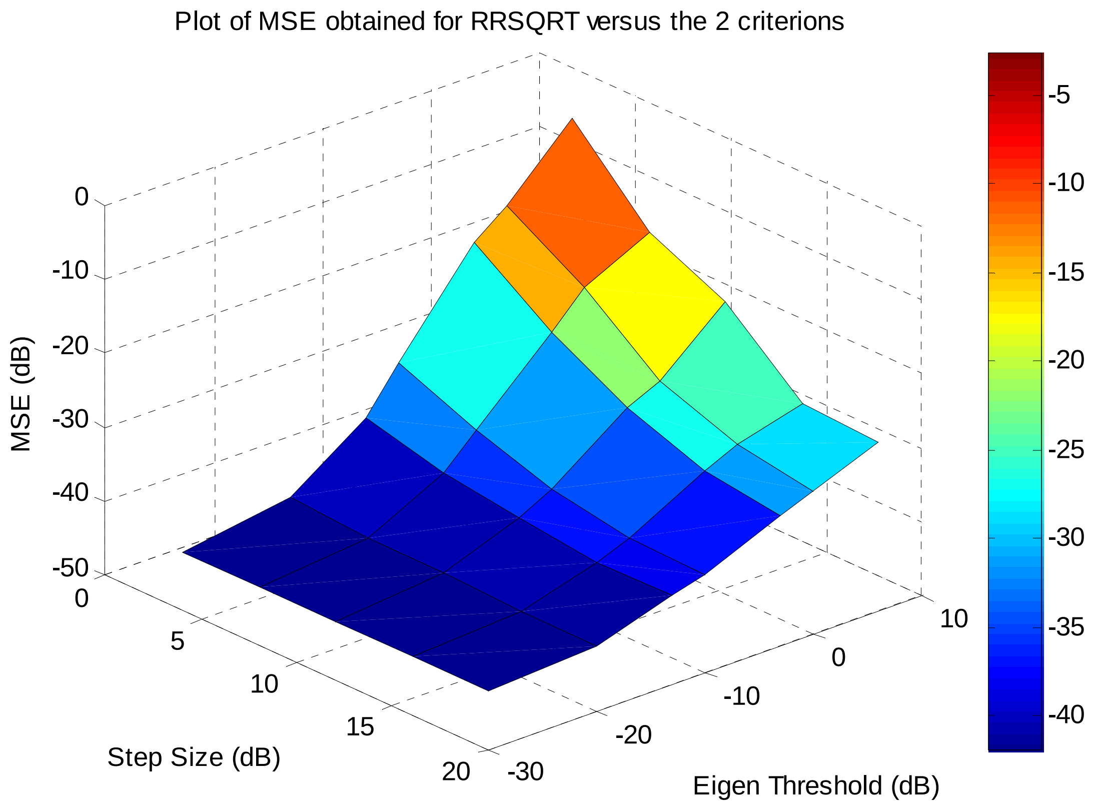


Figure 3-8: MSE (dB) obtained for RRSQRT

Now, by examining Table 3.6, we can see that the final MSE obtained can vary greatly from as bad a result of -2.239 dB to the full rank SRCF result of -42.216 dB depending on the combination of the two criterions. When the threshold of criteria 2 is set too low, i.e. 0.001% of the initial Eigen value, the RRSQRT does not experience any reduction in rank and thus its results' accuracy is equivalent to the full rank SRCF. This phenomenon is verified by the results shown in Table 3.7. On the other hand, if the threshold of criteria 2 is set as high as 4% of the initial Eigen value, many of the dominant Eigen vectors are too hastily discarded such that much information is lost when creating $\tilde{\mathbf{S}}(l/l)$ in the measurement-step process. This phenomenon can again be verified by the small number of remaining dominant Eigen vectors in Table 3.7 corresponding to the 4 % Threshold. As a result, the final MSE obtained from using a high threshold of criteria 2 is bad. As for the different step-size variation in the expected value of MSE used in criteria 1, it is observed from the results that it is better to use a larger step-size for triggering the Eigen decomposition and reduction-step processes.

Moreover, by applying the 2 criterions properly as seen in row 2 and row 3 of Table 3.6, we can see that the RRSQRT can achieve good results with little error or low MSE even when the rank is reduced substantially from 961 to as low as 257 as seen in Table 3.7. This is indeed a surprising phenomenon. In fact, when the final rank has been reduced to a value of 40 as seen in the combination of the 20 dB Step size for criteria 1 and 0.05% Threshold for criteria 2, we can still achieve a reasonably low MSE of –38.148 dB. Thus, this simulation has proved successfully that Verlaan's RRSQRT can also be used for the SAR image formation of our radar model.

Table 3.7: Remaining Eigen vectors for RRSQRT simulation

% of initial Eigen Value used for Criteria 2	Step size in dB for MSE variation used for Criteria 1				
	4	8	12	16	20
0.001	961	961	961	961	961
0.01	230	245	285	313	257
0.05	26	35	54	89	40
0.1	9	22	29	49	16
0.5	1	5	12	21	3
1.0	1	2	4	11	2
4.0	1	1	3	1	1

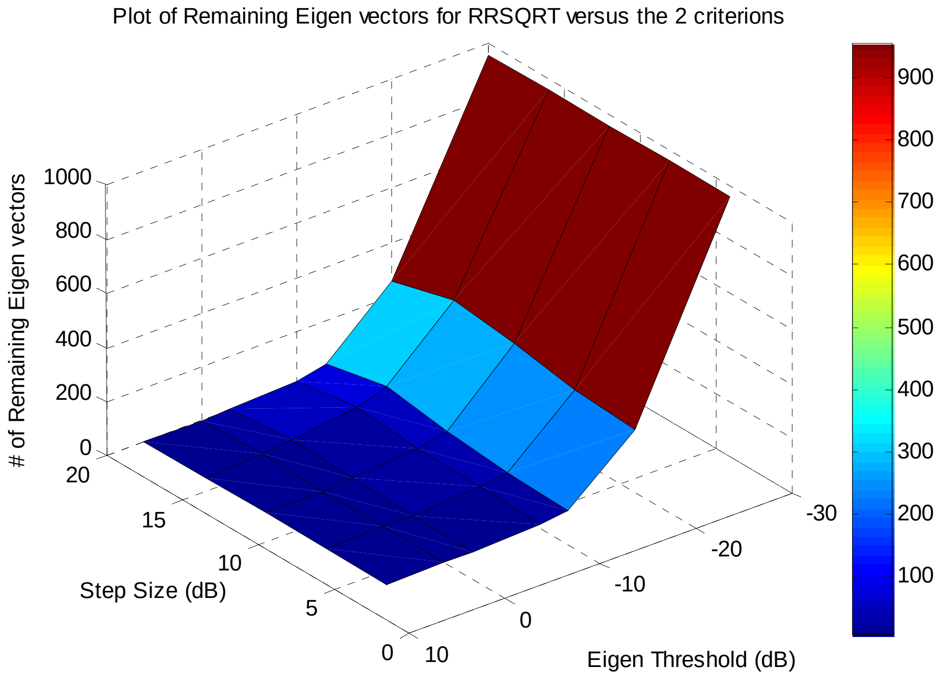


Figure 3-9: Remaining Eigen Vectors obtained for RRSQRT

Table 3.8: Results of Computational Time (sec) for RRSQRT simulation

% of initial Eigen Value used for Criteria 2	Step size in dB for MSE variation used for Criteria 1				
	4	8	12	16	20
0.001	289.275	168.691	120.879	95.145	96.552
0.01	78.994	64.331	60.305	58.405	57.667
0.05	48.978	46.594	47.052	49.748	49.500
0.1	45.493	44.583	45.882	48.595	48.125
0.5	41.804	41.951	44.801	47.562	47.236
1.0	41.984	42.084	44.162	46.843	47.000
4.0	41.201	41.586	43.992	46.604	46.853

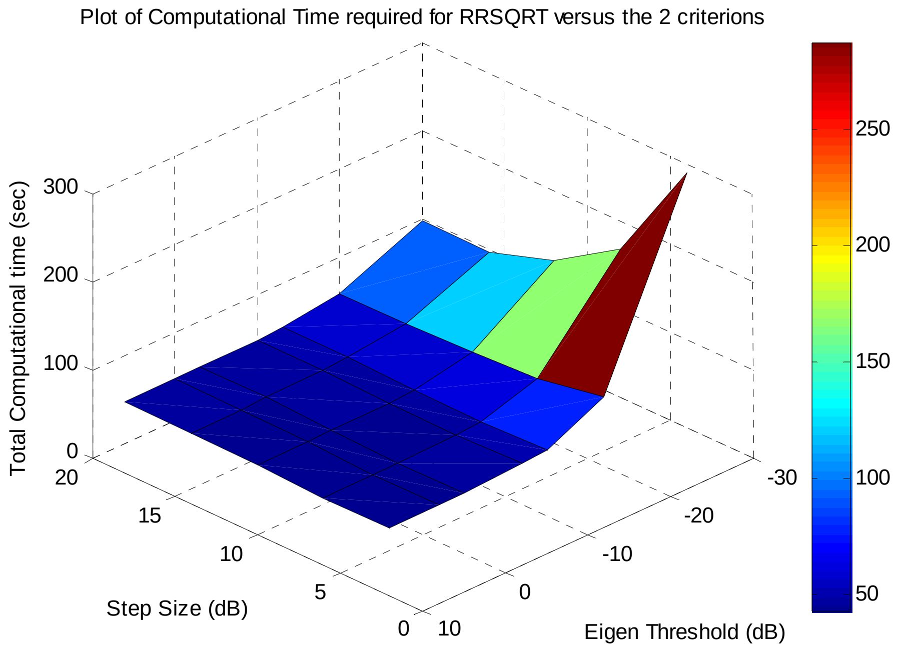


Figure 3-10: Total Computational Time obtained for RRSQRT

Next, by looking at Table 3.8, we can see that when there is no reduction of rank in the RRSQRT as in row 1 of Table 3.8, the total computational time is very much higher than when there is significant rank reduction as in row 3 to row 7 of Table 3.8. This outcome is due to the huge computational time required by the Eigen decomposition for a full rank matrix of 961 × 961 as compared to a smaller reduced rank matrix. Furthermore, when the step-size for invoking the Eigen decomposition is small, i.e. 4 dB or 8 dB, more Eigen decomposition calls will be invoked in those test scenarios. In order to verify this observation, the simulation is rerun and the computational time required for all the Eigen decomposition operations is then subtracted from the total computational time as shown in Table 3.8. The new results are then tabulated in Table 3.9 below:

Table 3.9: Computational Time (sec) minus Eigen decomposition for RRSQRT

% of initial Eigen Value used for Criteria 2	**Step size in dB for MSE variation used for Criteria 1**				
	4	8	12	16	20
0.001	80.454	63.785	58.097	52.691	53.348
0.01	35.367	34.894	34.945	35.264	34.432
0.05	24.203	25.734	26.521	28.857	28.562
0.1	22.540	24.394	25.772	27.876	27.390
0.5	20.444	22.357	24.957	26.906	26.548
1.0	20.686	22.521	24.318	26.235	26.375

% of initial Eigen Value used for Criteria 2	**Step size in dB for MSE variation used for Criteria 1**				
	4	8	12	16	20
4.0	20.185	22.133	24.148	25.979	26.212

From Table 3.9, we can see that the computational time has reduced drastically when the Eigen decomposition timing is not taken into account. Using the Kalman filter bench mark timing of 37.206 sec as shown in Table 3.1, we can say that the criterions for the highlighted portions in Table 3.9 and Table 3.6 are able to produce acceptably good results with small MSE as well as low computational timing if the Eigen decomposition timing is not taken into account. Thus, if we can find a much more efficient method to perform the Eigen decomposition as compared to what is currently provided in MATLAB under the function “eig”, then it may be possible for the total computational timing in the highlighted portions of Table 3.8 to be less than the Kalman filter timing while providing acceptable results.

Next, having performed a thorough investigation of the performance of the full rank SRCF and the RRSQRT in this Chapter, I will move on to another type of filtering technique that is also found to be more efficient than the MMSE or Wiener filter in its implementation while providing the same level of accuracy. At the same time, the inherent features or structure of the new filter allows it to be easily implemented in a reduced rank fashion that is essential for my research. This new type of filter is known as the Multi-Stage Wiener Filter (MSWF) that is based on successive orthogonal projections of the original measurement data. In the next Chapter, I will discuss the equations pertaining to the standard MSWF as well as the results obtained from the MSWF on the SAR image formation problem for the non-uniformly distributed aperture radar system.

CHAPTER 4: THE MULTI-STAGE WIENER FILTER

4.1 BRIEF BACKGROUND OF THE MULTI-STAGE WIENER FILTER

In the previous Chapter, we have examined one efficient implementation of the MMSE or Wiener filter in the form of the Square Root Covariance filter (SRCF) and the Reduced Rank Square Root filter (RRSQRT). In this Chapter, we will look into another efficient implementation of the MMSE filter and this implementation is known as the Multi-Stage Wiener Filter or MSWF or short. This filter was first developed and presented in 1997 by J. Scott. Goldstein and Irving S. Reed under [4] and elaborated in much more details under [5]. Prior to the presentation of [4], J. Scott. Goldstein and Irving S. Reed had worked on issues related to reduced rank adaptive filtering and had also presented a paper under [16] in which the cross-correlation vectors of the measurement data correlation matrix **R** are used to reduce the rank of the matrix **R**. This method is known as the cross-spectral method and it is shown to be much more efficient than the well known principal component method (PC) that uses the dominant Eigen vectors of **R** to perform rank reduction. Subsequently, the multi-stage Wiener filter was introduced and it also makes use of the cross-correlation vector of the desired signal in its implementation although this filter is very different from the cross-spectral method.

Now, one big advantage of the newly developed MSWF is that the traditional matrix inverse operation $\mathbf{R}^{-1}$ required by the MMSE or Wiener filter is not needed in the MSWF implementation. Thus, this will translate to better computational performance. Also, due to its ease of implementation and low complexity as compared to other Eigen decomposition-based methods of filtering like the PC method, the MSWF has been greatly adopted by the communication community in many applications such as in interference suppression in Direct Sequence Code Division Multiple Access (DS-CDMA) communication systems as seen in [17] and [18] etc. In recent years, there has also been interest in applying the MSWF in various radar applications like the Space Time Adaptive Filtering (STAP) domain etc. Thus, this filter is also chosen to be investigated for its performance in the SAR image formation problem that is the focus of my thesis research. Next, after having given a brief background of the multi-stage Wiener filter, I will provide the details and equations of this filter in the following section.

4.2 THE GENERIC MULTI-STAGE WIENER FILTER

The Multi-Stage Wiener Filter (MSWF) consists of a series of stage-by-stage decomposition of the initial measurement data using orthogonal projections. In terms of the rank of the filter, each stage of decomposition is equivalent to the increase of the filter processing by an additional rank of one. Therefore, for a Wiener filter of rank *N*, the MSWF will also achieve full rank processing when it has performed *N* stages of decompositions. Thus, this will also mean that for the MSWF, reduced rank processing can easily be obtained by simply truncating the stages of decomposition before *N* stages are reached. Now, from the results presented in [18] and [19], it has been shown that the MSWF is able to obtain the full rank performance of the Wiener filter in terms of accuracy by just performing a small number of stages of decomposition that is much lesser than the value N. Thus, it allows the MSWF to outperform the other Eigen-based reduced rank filters like the principal component or cross-spectral method when the rank reduction is significant.

Next, the multi-stage Wiener filter can also be viewed as a way of decomposing the standard MMSE filter using a multi-resolution approach [20], in which each stage of decomposition is based on the statistical importance of the residual correlations from an innovation process. In this approach, each stage of decomposition is determined using a matched filter type of criterion, in which the 1st stage of decomposition will maximize the cross-correlation between the scalar desired signal and the initial measurement data vector. The subsequent stage of decomposition, recognizing that residual correlation between the desired signal and the unwanted signals are still present after the 1st stage of decomposition, then provides a matched filter criterion again after using the null space or orthogonal space of the cross-correlation vector in the preceding stage to form the new transformed measurement data vector. In this manner, a point will be reached in which all the significant correlations between the desired signal and the unwanted signals will be extracted and then finally removed from the output of the 1st stage of decomposition such that the end result is equivalent to the MMSE or Wiener filter. To provide a clearer picture of the above explanation, the structure of the MSWF for a scalar desired signal is provided in the following figure so that it is easier to follow through each stage of decomposition starting from the initial measurement data.

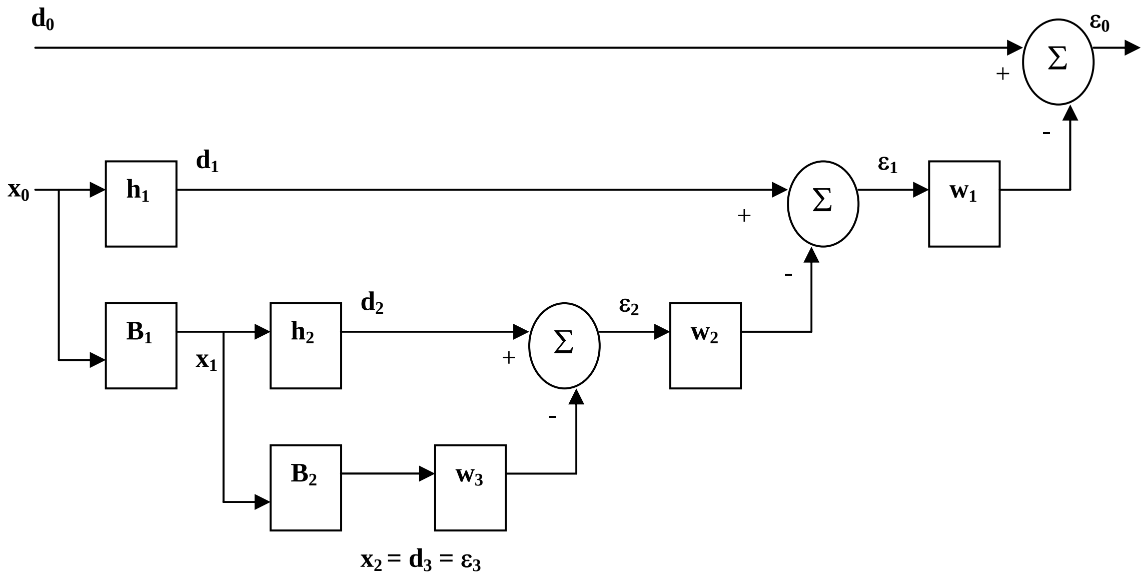


Figure 4-1: The structure of the Multi-Stage Wiener Filter for *N*=3

Note that in the above figure, the structure is for a MSWF with 3 stages of decomposition (*N*=3). Also, in the above figure, $\mathbf{x_0}$ is equivalent to the initial measurement data vector and d_0 is the scalar desired signal that is zero mean, Wide Sense Stationary (WSS) and is also a complex random number. As for the filters $\mathbf{h_i}$, they are the normalized cross correlation vectors between $\mathbf{x_i}$ and the scalars d_i and for the filters $\mathbf{B_i}$, they are the blocking matrices that eliminates the signal components in the direction of $\mathbf{h_i}$ such that $\mathbf{B_i h_i} = 0$. Thus, in figure 4.1, the value d_1 correspond to the maximum correlation between the desired signal and the input measurement data vector. The residual correlations that will need to be removed from d_1 are then obtained as d_2, d_3 etc and subsequently removed from d_1 such that the final estimate for the desired scalar signal d_0 is given by the equation:

$$\hat{d}_0 = w_1^* \varepsilon_1 \tag{4.1}$$

Notice that in the above figure 4.1, there appears to be a forward iteration and a backward iteration step in the computation of the various parameters. In fact, in the implementation of the multi-stage Wiener filter, there are 3 steps associated with it, namely, the forward iteration step, the turn around step and the backward iteration step. These steps are explained in great details in [5] and summarized again in [21] with relevance to the radar model. As for the equations pertaining to each of the step, they are shown below and will be used for implementation of the MSWF in the radar model for the estimation of each scattering coefficient γ_ι of the SAR image:

- Step 1: Forward Iteration for *i* = 1 to *N*-1

$$
\begin{aligned}
\delta_i &= \sqrt{\mathbf{r}_{x_{i-1}d_{i-1}}^{\mathrm{H}} \mathbf{r}_{x_{i-1}d_{i-1}}} \\
\mathbf{h}_i &= \frac{\mathbf{r}_{x_{i-1}d_{i-1}}}{\delta_i} \\
d_i &= \mathbf{h}_i^{\mathrm{H}} \mathbf{x}_{i-1} \\
\sigma_{d_i}^2 &= \mathbf{h}_i^{\mathrm{H}} \mathbf{R}_{x_{i-1}} \mathbf{h}_i \\
\mathbf{B}_i &= null\{\mathbf{h}_i\} \\
\mathbf{x}_i &= \mathbf{B}_i \mathbf{x}_{i-1} \\
\mathbf{R}_{x_i} &= \mathbf{B}_i \mathbf{R}_{x_{i-1}} \mathbf{B}_i^{\mathrm{H}} \\
\mathbf{r}_{x_i d_i} &= \mathbf{B}_i \mathbf{R}_{x_{i-1}} \mathbf{h}_i
\end{aligned}
\qquad (4.2)
$$

- Step 2: Turn-around at *i* = *N*

This step is special in that it can be interpreted as either the termination step of the forward iteration or as the 1st of the backward iteration step. The equations pertaining to the "Turn-around" step are as follows:

$$
\begin{aligned}
\delta_N &= \sqrt{\mathbf{r}_{x_{N-1}d_{N-1}}^{\mathrm{H}} \mathbf{r}_{x_{N-1}d_{N-1}}} \\
\mathbf{h}_N &= \frac{\mathbf{r}_{x_{N-1}d_{N-1}}}{\delta_N} \\
d_N &= \mathbf{h}_N^{\mathrm{H}} \mathbf{x}_{N-1} \\
\sigma_{d_N}^2 &= \mathbf{h}_N^{\mathrm{H}} \mathbf{R}_{x_{N-1}} \mathbf{h}_N \\
\xi_N &= \sigma_{d_N}^2 \\
w_N &= \frac{\delta_N}{\xi_N} \\
\varepsilon_N &= d_N
\end{aligned}
\tag{4.3}
$$

- Step 3: Backward Iteration for *i* = *N*-1 to 1

In the backward iteration step, for index *i* beginning with *i* = *N*-1 and ending with *i* = 1 in decrement step size of 1, the equations are as follows:

$$
\begin{aligned}
\xi_i &= \sigma_{d_i}^2 - \frac{|\ \delta_{i+1}\ |^2}{\xi_{i+1}} \\
w_i &= \frac{\delta_i}{\xi_i} \\
\varepsilon_i &= d_i - w_{i+1}^* \varepsilon_{i+1}
\end{aligned}
\tag{4.4}
$$

Finally, the estimate of the desired signal is given by equation 4.1 and the error variance or MSE of the estimate $\hat{d}_0$ is given by the expression:

$$
\begin{aligned}
\sigma_{\varepsilon_0}^2 &= \sigma_{d_0}^2 - \sigma_{\hat{d}_0}^2 \\
&= \sigma_{d_0}^2 - w_1^* \xi_1 w_1
\end{aligned}
\tag{4.5}
$$

Also, notice the similarity between equation 4.5 and equation 2.49 of [10] that is for the MMSE of the standard Wiener filter. At this point, there are still the issues of the forming of the blocking matrices $\mathbf{B_i}$ and the definition of the initial conditions before the MSWF implementation can be carried out. The issue of forming the blocking matrices will first be discussed below:

4.2.1 FORMING THE BLOCKING MATRIX **B**

Now, in the previous section, the blocking matrix $\mathbf{B_i}$ is defined to be the null space of the normalized cross correlation vector $\mathbf{h_i}$ such that $\mathbf{B_i h_i} = 0$. In [5], the authors have proposed 3 algorithms of forming the blocking matrix $\mathbf{B_i}$. Basically the first two algorithms consist of using either a Singular Value Decomposition (SVD) or a QR decomposition to obtain a unitary blocking matrix and they are described in [22]. The equations for the 2 algorithms are as follows:

- Algorithm 1: Using SVD decomposition

$$\begin{aligned} [\mathbf{U},\mathbf{S},\mathbf{V}] &= \mathrm{svd}(\mathbf{h_i^{\mathrm{H}}}) \\ \mathbf{B_i} &= [\mathbf{V}(:,2:N)]^{\mathrm{H}} \end{aligned} \tag{4.6}$$

- Algorithm 2: Using QR decomposition

$$\begin{aligned} [\mathbf{Q},\mathbf{R}] &= \mathrm{qr}(\mathbf{h_i^{\mathrm{H}}}) \\ \mathbf{B_i} &= [\mathbf{Q}(:,2:N)]^{\mathrm{H}} \end{aligned} \tag{4.7}$$

Note that if the dimension N of the cross correlation vector $\mathbf{h_i}$ is huge; both these algorithms will be very computationally expensive. Besides the 2 algorithms above, the authors of [5] also propose a 3rd algorithm in Appendix A of the same paper that is supposedly less complex than the first two algorithms but will generate a non-unitary blocking matrix instead of a unitary blocking matrix. Although the 3rd algorithm is less computationally expensive than the first two algorithms, it still takes some time to generate if the value of *N* is large. Also, another point to note is that for the 3 algorithms mentioned above, the blocking matrix formed is rectangular and thus the column space of the input measurement data vector will always be reduced by 1 whether the blocking matrix is applied to the data vector.

As mentioned above, all 3 algorithms proposed above are not very efficient when used for generating the blocking matrix $\mathbf{B_i}$. On top of that, as the output measurement vector $\mathbf{x_i}$ from the blocking matrix $\mathbf{B_i}$ is different in dimension from the input measurement vector $\mathbf{x_{i-1}}$, thus this will result in additional hardware or software complexity in the implementation to account for all these vectors of different lengths. Thus, to overcome the problem, it was decided to use the orthogonal projection of $\mathbf{h_i}$ to form the square blocking matrix instead such that:

$$\mathbf{B_i} = \mathbf{I} - \mathbf{h_i}\mathbf{h_i}^{\mathrm{H}} \tag{4.8}$$

By looking at equation 4.8, we can see that it is now a very simple operation to form the blocking matrix. At the same time, this will also improve the computation of $\mathbf{x_i}$ from $\mathbf{x_{i-1}}$ since the new way of computation is:

$$\begin{aligned} \mathbf{x}_i &= \mathbf{B}_i\mathbf{x}_{i-1} \\ &= [\mathbf{I} - \mathbf{h}_i\mathbf{h}_i^{\mathrm{H}}]\mathbf{x}_{i-1} \\ &= \mathbf{x}_{i-1} - \mathbf{h}_i d_i \end{aligned} \tag{4.9}$$

Interestingly, after verifying that this implementation is working properly, I also discovered subsequently that the same idea was proposed by [23] citing the same benefits as above. Thus, this reconfirms the decision to use equation 4.8 to form the blocking matrix.

4.2.2 DEFINING THE INITIAL CONDITIONS FOR MSWF

In this section, the initial conditions for implementing the MSWF will be defined [21]. From equation 4.2, we can see that it is required to define the initial matrix $\mathbf{R}_{\mathbf{x}_0}$ and the cross correlation vector $\mathbf{r}_{\mathbf{x}_0 d_0}$ before the forward iteration step can proceed. Now, the initial measurement data vector $\mathbf{x}_0$ is equivalent to the response vector **r** in my radar model. Therefore using equation 2.17, we can now define the following:

$$\begin{aligned} \mathbf{R}_{\mathbf{x}_0} &= \mathbf{P}\mathbf{R}_{d_0}\mathbf{P}^{\mathrm{H}} + \mathbf{K}_{\mathbf{n}} \\ \mathbf{K}_{\mathbf{n}} &= \sigma_{\mathbf{n}}^2 \mathbf{I} \\ \mathbf{R}_{d_0} &= \mathbf{K}_{\gamma} \\ &= \sigma_{\gamma}^2 \mathbf{I} \end{aligned} \tag{4.10}$$

In equation 4.10 above, the Gaussian measurement noise covariance matrix is $\mathbf{K}_\mathrm{n}$ and $\sigma_{\gamma}^2 = \mathrm{E}[\gamma_i^H \gamma_i]$ is the expected value of the square of the scattering coefficient of each image resolution cell or scatterer that has a mean value of zero. Also, for the MSWF implementation defined above, it is to be used for scalar signal estimation. Now, since there are many scattering coefficients in the SAR image formation problem, therefore, in order to estimate the value of the i^{th} scattering coefficient γ_i using the MSWF, the value of $\mathbf{r}_{\mathbf{x}_0 d_0}$ is equal to:

$$\mathbf{r}_{\mathbf{x}_0 d_0} = \mathbf{P}\mathbf{R}_{d_0}\mathbf{e}_i \tag{4.11}$$

in which $\mathbf{e}_i$ is a vector whose elements are all zeros except the element *i* that has a value of 1. As an example, if *i* = 5, then:

$$\mathbf{e}_5 = [0, 0, 0, 0, 1, 0, \ldots..]^T \qquad (4.12)$$

Thus, with the definition of equation 4.11, all the initial conditions for the MSWF implementation have been defined and it is now possible to start the MSWF stages of decomposition. In the next section, the performance of the MSWF of filter order $O(5\times M^2)$used to estimate the value of one scattering coefficient will be evaluated with that of the MMSE filter as well as with the Kalman filter.

4.3 PERFORMANCE OF MSWF FOR SCALAR ESTIMATION OF γ_i

In this section, a simulation is carried out using the experimental conditions as defined in Table 3.1 in Chapter 3 that uses a measurement size of 3060 to estimate a total of 961 image resolution cells. Also, the value of image resolution cell #1 or γ_1 is chosen as the desired signal to be estimated by the MSWF. By varying the stages of decomposition of the MSWF, the results obtained for the normalized value of the Expected Error Variance $\bar{\sigma}^2_{\varepsilon_0}$ and normalized value of the Computed Error Variance of γ_1 are then recorded and shown in Table 4.2 below. Note that the 2 parameters are defined as follows with N_t being the total number of resolution cells:

$$\bar{\sigma}^2_{\varepsilon_0} = \frac{\sigma^2_{\varepsilon_0}}{(\hat{\gamma}^H\hat{\gamma}/N_t)} \qquad (4.13)$$

$$\mathrm{Computed}\,MSE = \frac{(\hat{\gamma}(1) - \gamma(1))^H(\hat{\gamma}(1) - \gamma(1))}{(\hat{\gamma}^H\gamma/N_t)} \qquad (4.14)$$

Table 4.1: Results from MMSE and Kalman filter for image resolution cell #1

Filter Type	Total Time /sec	$\overline{\sigma}_{\varepsilon_0}^2$ /dB	Computed MSE /dB
Kalman	37.506	-40.475	-40.953
Wiener	220.711	-40.475	-40.953

Table 4.2: Results from MSWF for image resolution cell #1

Number of Stages	Total Time /sec	$\overline{\sigma}_{\varepsilon_0}^2$ /dB	Computed MSE /dB
1	40.782	-3.772	-4.389
2	43.613	6.840	-8.536
3	46.647	9.642	-7.023
4	49.713	-11.900	-17.022
5	52.746	-14.043	-32.492
6	55.749	-16.188	-16.110
7	58.753	-17.945	-24.852
8	61.802	-19.474	-22.048
9	64.914	-20.941	-17.414
10	67.839	-22.356	-19.327
20	98.187	-39.019	-36.650
25	113.110	-40.382	-40.513
30	128.404	-40.469	-40.865
35	143.064	-40.475	-40.947
40	158.797	-40.475	-40.964

Firstly, by looking at the results of Table 4.1, it is not surprising to see that both the Wiener filter and the Kalman filter producing the same results but with different computational timing. Secondly, from the MSE results shown in Table 4.2, some characteristics of the MSWF can be observed. As mentioned in the previous sections, each additional stage of decomposition of the MSWF is equivalent to the increase of the filter processing by an additional rank of one. Therefore, for a MSWF with number of decomposition stages equal to 1, it will correspond to only processing the filter with a reduced rank equal to 1. As such, the MSE results obtained are very poor compared to the full rank processing results. However, as the stages of decomposition increase, the performance of the MSWF also improves as seen by the corresponding MSE results. Eventually, it can be seen that the MSWF has obtained full rank MMSE filter result at a decomposition stage of about 40 which takes lesser computational time than the MMSE filter. Therefore, at this point, it seems that the MSWF is functioning properly in its estimation of the scattering coefficient γ_1 and using a small number of decomposition stages that is much less than the full rank of 961 for the matrix $\mathbf{R}_{\mathbf{x}_0}$.

Also, note that in obtaining the total computational time of both the Wiener filter and the MSWF, it includes the time required to form the initial condition matrix $\mathbf{R}_{\mathbf{x}_0}$ which takes 40.610 sec. As for the Kalman filter, as it is only using a subset of the measurements in each iteration step, thus the time needed for its initial condition preparation is negligible. Therefore, if it is assumed that the initial conditions are already present before using the filters, the modified timing is as shown in Table 4.3 and Table 4.4:

Table 4.3: Results from MMSE and Kalman filter with modified timing

Filter Type	**Total Time /sec**	$\overline{\sigma}_{\varepsilon_0}^2$ **/dB**	**Computed MSE /dB**
Kalman	37.506	-40.475	-40.953
Wiener	180.101	-40.475	-40.953

Table 4.4: Results from MSWF with modified timing

Number of Stages	**Total Time /sec**	$\overline{\sigma}_{\varepsilon_0}^2$ **/dB**	**Computed MSE /dB**
1	0.172	-3.772	-4.389
2	3.003	6.840	-8.536
3	6.037	9.642	-7.023
4	9.103	-11.900	-17.022
5	12.136	-14.043	-32.492
10	27.229	-22.356	-19.327
20	57.577	-39.019	-36.650
25	72.500	-40.382	-40.513
30	87.794	-40.469	-40.865
35	102.454	-40.475	-40.947
40	118.187	-40.475	-40.964

Looking at Table 4.3 and 4.4, we can observe that although the MSWF estimator is able to outperform the MMSE or Wiener filter in terms of computational time, it is still not as fast as the Kalman filter itself.

Next, after I have looked at the performance of the MSWF on its estimation of the scalar value of γ_1, the next experiment will be to repeat the same experiment for 960 times corresponding to the remaining 960 targets in the SAR image to ensure that the remaining targets will also converge at the decomposition stage of 40. Also, as the MSWF discussed so far can only estimate one scalar signal at a time, I will introduce the notation "scalar MSWF" at this point to distinguish it from the subsequent implementation of MSWF that can estimate more than 1 signal at a time. At the same time, to check the overall performance of the results of all the targets, two evaluation methods will be carried out. The 1st evaluation method will be to use the average expected error variance and the average computed error variance across all targets as the performance parameters where these parameters are defined as:

$$\bar{\sigma}^2_{\varepsilon_0} = \frac{\sum\limits_{i=1}^{i=N_t} \sigma^2_{\varepsilon_{0i}}}{(\hat{\gamma}^{\mathrm{H}}\hat{\gamma})} \tag{4.15}$$

$$\mathrm{Computed}MSE = \frac{(\hat{\gamma}-\gamma)^{\mathrm{H}}(\hat{\gamma}-\gamma)}{(\hat{\gamma}^{\mathrm{H}}\gamma)} \tag{4.16}$$

Note that there are some differences between equations 4.15 to 4.16 for all targets and equations 4.13 to 4.14 for a single target. As for the 2nd evaluation method, essentially it is to obtain the error vector ε_1 between the estimated values $\hat{\gamma}$ and the actual values γ for the MMSE filter as well as the error vector ε_2 for that of the

MSWF. A correlation analysis between these two error vectors is then performed using the inner product rule as follows:

$$\sigma_{\varepsilon_1\varepsilon_2} = \boldsymbol{\varepsilon}_1^* \boldsymbol{\varepsilon}_2 \big/ |\boldsymbol{\varepsilon}_1||\boldsymbol{\varepsilon}_2| \tag{4.17}$$

In equation 4.17, $\sigma_{\varepsilon_1\varepsilon_2}$ is the correlation measure between the two error vectors. If the two error vectors are identical, then $\sigma_{\varepsilon_1\varepsilon_2}$ will have a value that will be equal to 1. This will imply that the two vectors are perfectly correlated. However, if the two error vectors are non-identical, then $\sigma_{\varepsilon_1\varepsilon_2}$ will be less than 1. Using these two methods, the simulation is carried out for the 2nd experiment and the results are computed and shown below:

Table 4.5: Performance of MMSE, Kalman and scalar MSWF using 1st method

Filter Type	$\bar{\sigma}^2_{\varepsilon_0}$ **/dB**	**Computed MSE /dB**
Kalman	-41.933	-42.216
Wiener	-41.933	-42.216
MSWF	-41.933	-42.216

From Table 4.5, we can see that the scalar MSWF is able to achieve the same performance in terms of accuracy as the other two filters for all the 961 targets. To further verify this observation, the 2nd method using correlation measure is carried out and the results for the correlation measure $\sigma_{\varepsilon_1\varepsilon_2}$ turns out to be equal to 1 as well. Thus, these two methods prove that the scalar MSWF is functioning properly for the SAR image formation problem. At the same time, some plots showing the elements of the two error vectors ε_1 and ε_2 are provided to illustrate the perfect correlation between these two vectors.

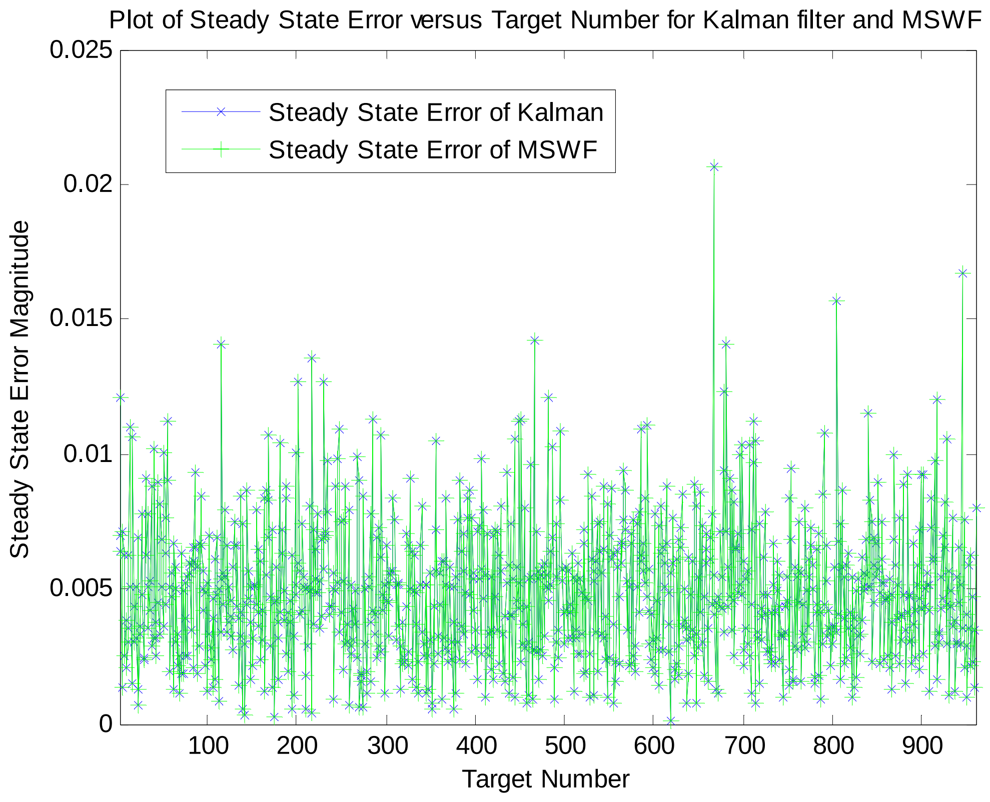


Figure 4-2: The Steady State Error magnitude across all targets

In Figure 4.2, we can see that the final error obtained from each image resolution cell or target has the same value from both the Kalman filter and the MSWF such that the two error curves coincide exactly with each other at all points. However, as there are too many targets in figure 4.2, certain subsets of the targets are re-plotted again so as to have a clearer picture of the correlation relationship between the two error vectors, i.e., the zoom-in version of Figure 4.2. These new plots are shown in figure 4.3 and figure 4.4 on the next page.

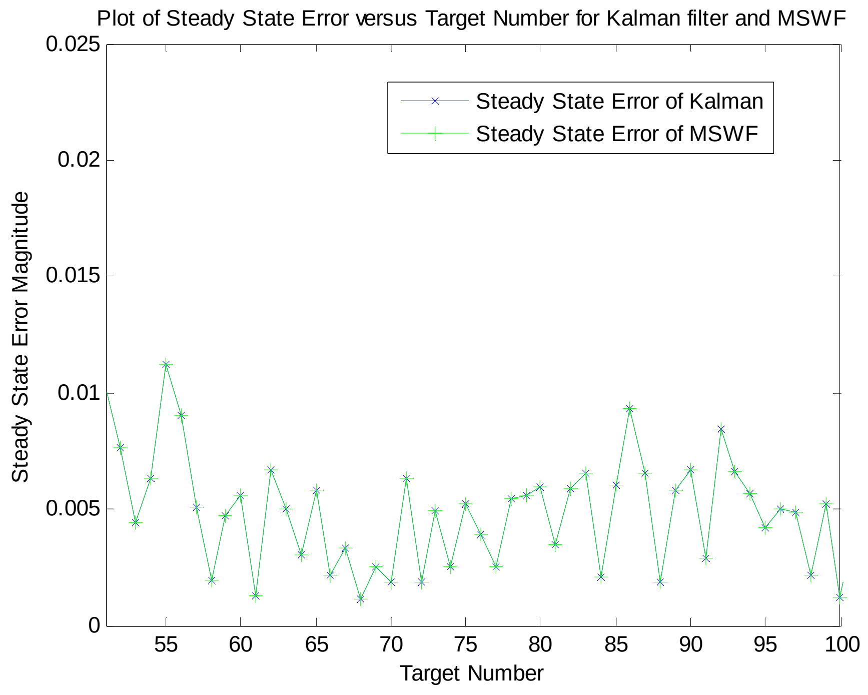


Figure 4-3: The Steady State Error magnitude from Targets 51 to 100

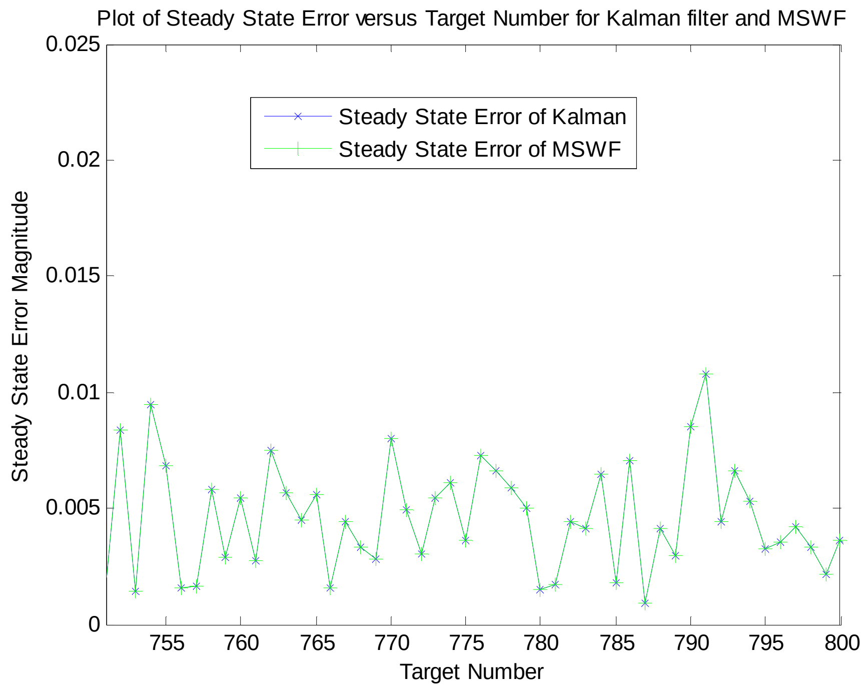


Figure 4-4: The Steady State Error magnitude from Targets 751 to 800

Now, the results and plots obtained above from the two evaluation methods have verified that the scalar MSWF is able to produce the final estimation results with equally good accuracies as with the full rank MMSE and Kalman filter while using less than full rank processing. However, to obtain the estimated scattering coefficients $\hat{\gamma}$ for the whole image of 961 resolution cells, it takes a total of 118.187*961 seconds (118.187 = time needed for 40 stages of decomposition per target) to complete the process if it is performed serially with one target at a time. On the other hand, to confine the time needed to process all targets to be the same as processing one target, (e.g., 118.187 seconds) it will require 961 computing machines to perform the tasks in parallel. Thus, neither of these scenarios are feasible from an implementation point of view. Therefore, to overcome the limitations posed by the scalar MSWF that can only estimate one signal at a time, the next step will be to search for a form of the MSWF that can estimate multiple signals or a vector of signals at a time. By using the MSWF in such a manner, i.e. vector MSWF, we can either reduce the total computational time for all targets from 118.187*961 seconds or reduce the amount of parallel machines needed from 961 etc. In the next section, the results of the search for the vector MSWF are discussed along with the details of implementation for such type of MSWF.

4.4 THE VECTOR MULTI-STAGE WIENER FILTER

In order to use the MSWF to estimate multiple signals or using the term "Multi-user Detection" or MUD in communication applications, researchers have been trying to satisfy this goal by using parallel processing with 1 output signal per processor as can be seen in [24] and [25]. However, as mentioned in the previous section, this approach will not be feasible if there are many signals to be detected or estimated. Looking back, it is briefly mentioned in [20] that the MSWF can be used for detecting multiple signals using a Multi-Stage Matrix Wiener Filter implementation along with some implementation details. The term "Matrix" in [20] refers to the facts that unlike scalar signal detection in which the weights of the MSWF will form a vector, i.e. $\mathbf{w_{mswf}} = [w_1, w_2, w_3 \ldots w_N]$, the weights of the MSWF for multiple signal detection will form a matrix $\mathbf{W_{mswf}}$ instead. Therefore, the notation "Multi-Stage Matrix Wiener Filter " used by [20] and my notation "vector MSWF" used in this thesis are referring to the same implementation. After further research, another paper [26] is found that also provides additional information for the vector MSWF implementation etc. Thus, using the information from the two papers, the equations for the vector MSWF of filter order $O(8\times(N_t^3)+8\times(N_t^2\times M)+5\times(M^2\times N_t))$ are slightly modified from equation 4.2 to 4.4 and are shown as follows:

- Step 1: Forward Iteration for i = 1 to N-1

$$\begin{aligned}
\boldsymbol{\delta}_i &= (\mathbf{R}^{\mathrm{H}}_{x_{i-1}d_{i-1}}\mathbf{R}_{x_{i-1}d_{i-1}})^{1/2}\\
\mathbf{H}_i &= \mathbf{R}_{x_{i-1}d_{i-1}}\boldsymbol{\delta}_i^{-1}\\
\mathbf{d}_i &= \mathbf{H}_i^{\mathrm{H}}\mathbf{x}_{i-1}\\
\mathbf{R}_{\mathbf{d}_i} &= \mathbf{H}_i^{\mathrm{H}}\mathbf{R}_{x_{i-1}}\mathbf{H}_i\\
\mathbf{B}_i &= null\{\mathbf{H}_i\}
\end{aligned} \tag{4.18}$$

$$
\begin{aligned}
\mathbf{x}_i &= \mathbf{B}_i\mathbf{x}_{i-1} \\
\mathbf{R}_{x_i} &= \mathbf{B}_i\mathbf{R}_{x_{i-1}}\mathbf{B}_i^{\mathrm{H}} \\
\mathbf{R}_{x_i\mathbf{d}_i} &= \mathbf{B}_i\mathbf{R}_{x_{i-1}}\mathbf{H}_i
\end{aligned}
$$

Note that in equation 4.18, the cross correlation vector $\mathbf{r}_{\mathbf{x}_0 d_0}$ as well as its normalized version $\mathbf{h_i}$ in equation 4.2 have been transformed from vectors to matrices $\mathbf{R}_{\mathbf{x}_0\mathbf{d}_0}$ and $\mathbf{H_i}$. At the same time, the normalizing value δ_i has also become a matrix $\boldsymbol{\delta}_i$ and the square root operation required to obtain $\boldsymbol{\delta}_i$ is not the standard scalar square-root function but the fast and accurate Cholesky decomposition operation instead. Also, all scalar division operations have been replaced by matrix inverse operations. As for the expected variance $\sigma_{d_i}^2$ of one target, it has been replaced by the Covariance Matrix $\mathbf{R}_{\mathbf{d}_i}$ of multiple targets.

- Step 2: Turn-around at $i = N$

$$
\begin{aligned}
\boldsymbol{\delta}_N &= (\mathbf{R}_{x_{N-1}d_{N-1}}^{\mathrm{H}}\mathbf{R}_{x_{N-1}d_{N-1}})^{1/2} \\
\mathbf{H}_N &= \mathbf{R}_{x_{N-1}d_{N-1}}\boldsymbol{\delta}_N{}^{-1} \\
\mathbf{d}_N &= \mathbf{H}_N^{\mathrm{H}}\mathbf{x}_{N-1} \\
\mathbf{R}_{\mathbf{d}_N} &= \mathbf{H}_N^{\mathrm{H}}\mathbf{R}_{x_{N-1}}\mathbf{H}_N \\
\boldsymbol{\xi}_N &= \mathbf{R}_{\mathbf{d}_N} \qquad (4.19) \\
\mathbf{W}_N &= \boldsymbol{\xi}_N{}^{-1}\boldsymbol{\delta}_N \\
\boldsymbol{\varepsilon}_N &= \mathbf{d}_N
\end{aligned}
$$

Note that in equation 4.19, the scalar weight w_N has been replaced by the weight matrix $\mathbf{W}_N$ instead.

- Step 3: Backward Iteration for *i* = *N*-1 to 1

$$\begin{aligned}
\boldsymbol{\xi}_i &= \mathbf{R}_{\mathbf{d}_i} - \boldsymbol{\delta}_{i+1}^{\mathrm{H}}\boldsymbol{\xi}_{i+1}\boldsymbol{\delta}_{i+1} \\
\mathbf{W}_i &= \boldsymbol{\xi}_i^{-1}\boldsymbol{\delta}_i \\
\boldsymbol{\varepsilon}_i &= \mathbf{d}_i - \mathbf{W}_{i+1}^{\mathrm{H}}\boldsymbol{\varepsilon}_{i+1}
\end{aligned} \tag{4.20}$$

Finally, the estimated values of all the scattering coefficients are given by the following expression:

$$\hat{\gamma} = \mathbf{W}_{\mathbf{1}}^{\mathrm{H}}\boldsymbol{\varepsilon}_1 \tag{4.21}$$

4.4.1 ADJUSTMENT OF DATA LENGTH DUE TO CHOLESKY FUNCTION

In the equations for the vector MSWF implementation, it is mentioned that the square root operation of a matrix is performed using the fast and accurate Cholesky decomposition that will produce an upper triangular matrix that is the square root of the input matrix, i.e. for a square matrix **X**, its square root **U** is obtained using:

$$\begin{aligned}
\mathbf{X} &= \mathbf{U}^{\mathrm{H}}\mathbf{U} \\
\mathbf{U} &= \mathrm{cholesky}(\mathbf{X})
\end{aligned} \tag{4.22}$$

Now, one requirement for performing Cholesky decomposition is that the input matrix must be positive definite which means that none of its Eigen values are zero. Otherwise, the Cholesky decomposition operation will fail completely.

With this knowledge of the Cholesky operation, we will now examine its usage in the vector MSWF implementation. As can be seen from equation 4.18 and 4.19, the Cholesky operation is used to compute the normalization matrix of the initial cross correlation matrix $\mathbf{R}_{\mathbf{x}_0\mathbf{d}_0}$ etc. Also, by referring to equation 4.11, we can see that the initial cross correlation matrix $\mathbf{R}_{\mathbf{x}_0\mathbf{d}_0}$ is formed from the **P** matrix defined in equation 2.16 that is equal to the combination of the normalized space-time receiver measurements or response vectors. However, due to the features of the radar model simulator that also accounts for targets that are at the near end or far end of the radar receiver platforms, there are many zeros in the initial and end rows of the **P** matrix. Unfortunately, these zeros will cause the initial Cholesky operation in the forward iteration to fail as it will result in the $\mathbf{R}_{\mathbf{x}_0\mathbf{d}_0}$ matrix becoming positive semi-definite instead. Thus, in order to implement the vector MSWF, the initial portion of the **P** matrix and the response vector **r** is discarded in the computing process.

4.4.2 MINIMUM DATA LENGTH RELATIONSHIP TO TARGET SIZE

For the scalar MSWF implementation, there is no requirement on the minimum length of the measurement data to be used when performing the stages of decomposition. However, for vector MSWF implementation, if there are *K* targets or image resolution cells to be estimated, then the length of the measurement data or response vector **r** cannot be less than *K*. Otherwise, this will result in the product $(\mathbf{R}_{\mathbf{x}_0 d_0}{}^{\mathrm{H}}\mathbf{R}_{\mathbf{x}_0 d_0})$ with a size of ($K \times K$) having some Eigen values that are equal to zero or becomes a positive semi-definite matrix. Subsequently, this will again result in the failure of execution of the Cholesky operation at the 1st step of the forward iteration.

4.4.3 DIAGONAL LOADING OF THE COVARIANCE MATRIX $\mathbf{R}_{\mathbf{d}_i}$

Besides truncating the length of the response vector **r** for the vector MSWF implementation, there is another issue that have arise as a result of using the Cholesky decomposition operation. By referring to equation 4.18 and 4.19, we can see that the covariance matrix $\mathbf{R}_{\mathbf{d}_i}$ is computed at every forward iteration step. However, due to finite computer precision, this may cause $\mathbf{R}_{\mathbf{d}_i}$ to change from a positive definite matrix to a positive semi-definite matrix due to rounding errors. Once this change occurs, the forward iteration will not be able to proceed with further stages of decomposition as the Cholesky operation will fail at the 1st step of the next stage of decomposition. Therefore, in order to overcome this shortcoming of the vector MSWF implementation, the solution is to apply diagonal loading to the covariance matrix $\mathbf{R}_{\mathbf{d}_i}$ during its computation. In this manner, this will ensure that $\mathbf{R}_{\mathbf{d}_i}$ will not become positive semi-definite as the iteration proceeds. As for the amount of diagonal loading to be introduced to $\mathbf{R}_{\mathbf{d}_i}$, after several trials, the value corresponding to 0.1% of the measurement noise power σ_n^2 used in the radar model. As a result of the introduction of diagonal loading, the computation of $\mathbf{R}_{\mathbf{d}_i}$ and $\mathbf{R}_{\mathbf{d}_N}$ is changed to the following expression:

$$\begin{aligned} \mathbf{DL} &= 0.001\sigma_n^2\mathbf{I} \\ \mathbf{R}_{\mathbf{d}_i} &= \mathbf{H}_i^{\mathrm{H}}\mathbf{R}_{x_{i-1}}\mathbf{H}_i + \mathbf{DL} \end{aligned} \tag{4.23}$$

Finally, after all these considerations are taken into account, the vector MSWF is then successfully implemented and its performance is evaluated in the next section.

4.5 PERFORMANCE OF MSWF FOR VECTOR ESTIMATION OF γ

In this section, a simulation is carried out using the experimental conditions as defined in Table 3.1 in Chapter 3 but however the measurement size has been reduced from 3060 to 2856 as is explained in section 4.4.1. Also, the complex scattering coefficients γ are also taken from the KU image. As a start, the simulation will first begin with estimating a target group containing only a single target and then followed by various increments of target or image numbers in the target group such that it will reach the total of 961 image resolution cells. At the same time, the value of image resolution cell #1 or γ_1 is chosen as the target whose performance is to be tracked within the group throughout the process. By varying the stages of decomposition of the vector MSWF, the results obtained for the normalized expected error variance $\bar{\sigma}^2_{\varepsilon_0}$ and the normalized computed error variance of γ_1 along with the average expected error variance and the average computed error variance across the few target group sizes are then recorded and the results of these target group sizes are shown in the following tables. Note that the average expected error variance and the average computed error variance across each target group size are defined as follows where *M* is the number of targets to be estimated:

$$\bar{\sigma}^2_{\varepsilon_0} = \frac{\frac{1}{M}\sum_{i=1}^{i=M} \sigma^2_{\varepsilon_{0i}}}{(\hat{\gamma}^{\mathrm{H}}\hat{\gamma}/N_t)} \tag{4.24}$$

$$\text{Computed}MSE = \frac{1}{M}\left(\frac{(\hat{\gamma}(1:M)-\gamma(1:M))^{\mathrm{H}}(\hat{\gamma}(1:M)-\gamma(1:M))}{(\hat{\gamma}^{\mathrm{H}}\gamma/N_t)}\right) \tag{4.25}$$

As with the scalar MSWF implementation, there is a time incurred of about 35.239 seconds needed to form the initial condition matrix $\mathbf{R}_{\mathbf{x}_0}$ for both the Wiener filter and the MSWF. In the following tables that are shown, this time is included in the total time incurred:

Table 4.6: Results from MMSE and Kalman filter across all Target resolution cells

Filter Type	**Total Time /sec**	$\bar{\sigma}_{\varepsilon_0}^2$ **/dB for Target 1**	**Computed MSE /dB for Target 1**	**Average** $\bar{\sigma}_{\varepsilon_0}^2$ **/dB**	**Average Computed MSE /dB**
Wiener	186.078	-41.021	-42.253	-41.101	-41.037
Kalman	35.562	-41.021	-42.253	-41.101	-41.037

Table 4.7: Results from vector MSWF for Target group size *M* = 1

Number of Stages	**Total Time /sec**	$\bar{\sigma}_{\varepsilon_0}^2$ **/dB for Target 1**	**Computed MSE /dB for Target 1**	**Average** $\bar{\sigma}_{\varepsilon_0}^2$ **/dB**	**Average Computed MSE /dB**
1	35.378	-4.431	-5.8382	-	-
2	37.856	-7.423	-20.512	-	-
3	40.411	-9.995	-23.824	-	-
4	43.268	-12.523	-23.037	-	-
5	46.221	-14.922	-19.714	-	-
10	58.667	-26.050	-25.405	-	-
20	84.485	-40.044	-37.379	-	-
30	109.627	-41.011	-42.124	-	-

Table 4.8: Results from vector MSWF for Target group size *M* = 10

Number of Stages	Total Time /sec	$\overline{\sigma}_{\varepsilon_0}^2$ /dB for Target 1	Computed MSE /dB for Target 1	Average $\overline{\sigma}_{\varepsilon_0}^2$ /dB	Average Computed MSE /dB
1	35.886	-4.578	-5.943	-4.426	-4.946
2	40.229	-7.922	-14.165	-7.889	-9.093
3	44.729	-10.693	-15.343	-10.924	-11.155
4	49.151	-13.528	-15.032	-13,762	-14.469
5	53.603	-16.159	-24.153	-16.451	-15.731
10	75.270	-29.947	-29.245	-29.113	-29.704
20	119.625	-40.972	-41.490	-40.697	-42.869
25	142.431	-41.016	-42.081	-40.759	-43.104

Table 4.9: Results from vector MSWF for Target group size *M* = 100

Number of Stages	Total Time /sec	$\overline{\sigma}_{\varepsilon_0}^2$ /dB for Target 1	Computed MSE /dB for Target 1	Average $\overline{\sigma}_{\varepsilon_0}^2$ /dB	Average Computed MSE /dB
1	39.134	-6.063	-4.508	-5.7795	-5.505
2	57.471	-10.524	-23.939	-10.364	-10.951
3	75.965	-14.615	-16.093	-14.598	-14.887
4	94.333	-18.843	-25.032	-18.897	-17.597
5	112.979	-23.604	-33.228	-23.561	-23.299
10	203.642	-41.017	-42.253	-40.939	-41.171

Table 4.10: Results from vector MSWF for Target group size *M* = 961

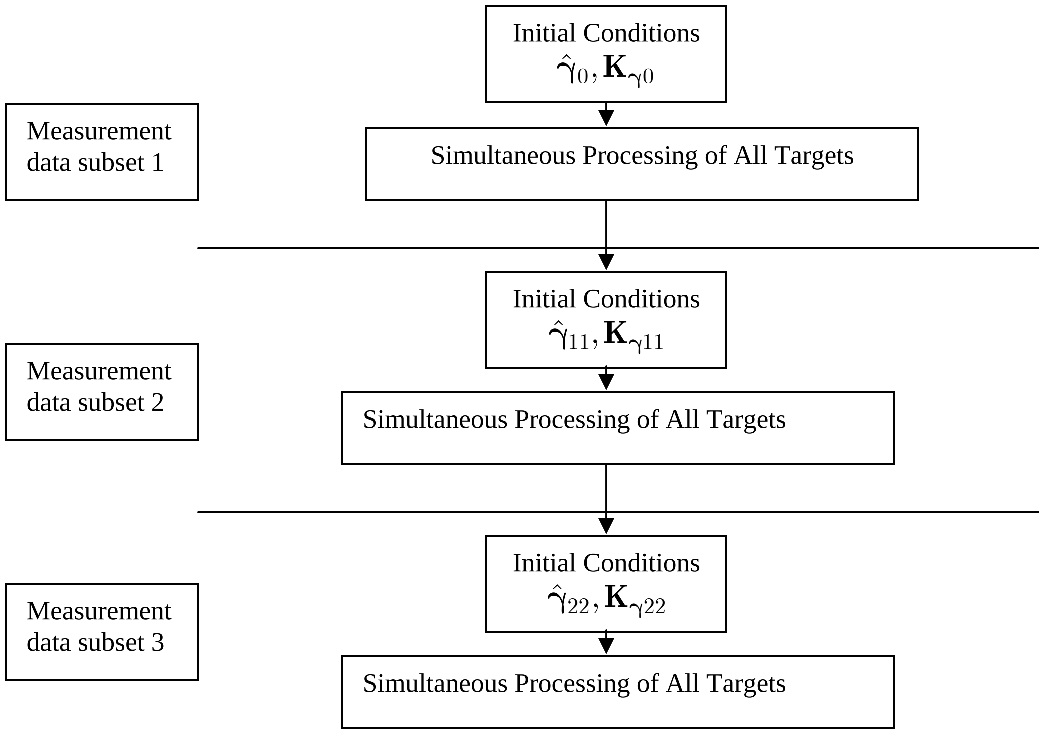
Number of Stages	Total Time /sec	$\overline{\sigma}_{\varepsilon_0}^2$ /dB for Target 1	Computed MSE /dB for Target 1	Average $\overline{\sigma}_{\varepsilon_0}^2$ /dB	Average Computed MSE /dB
1	112.572	-41.016	-42.253	-41.097	-41.037

From the results shown in the tables above, we can observe two trends. Firstly, we can see that as the number of targets within the group increases, it will lead to faster convergence of the targets using the vector MSWF in terms of stages of decomposition. Secondly, as the time needed for each stage of decomposition will take a longer time with an increase in the target group size, thus there is an overall increase in the computational time needed for convergence when the target size increases. However, when all the targets are used together in the decomposition as shown in Table 4.10, the trends reverse as all the targets achieved their steady state estimated values using just 1 stage of decomposition and in a shorter period of time. Also, from Table 4.10, we can see that the results obtained from the vector MSWF when using all targets together are the same as that of the Kalman or Wiener filter shown in Table 4.6. Thus, we can conclude that the vector MSWF is functioning properly in the estimation of the scattering coefficients γ.

At this point in time, after acquiring the capability of the MSWF to perform vector estimations of signals, the next important task will be to determine the choice of target group size that will allow all the targets in γ to reach their steady state estimated values in the shortest possible time given the available computing resources. Unlike Kalman or MMSE filter where it is not possible to divide the overall processing across various machines, it is now possible for the targets to be divided into groups and each group interdependently processed by a machine when using the MSWF. If enough machines are available such that all target groups can be processed simultaneously, this type of implementation will be known as the “Parallel MSWF” implementation. On the other hand, if there is only one machine available, it is still possible to perform the MSWF decomposition on each group in a serial manner with one group at a time. This type of implementation will be known as the “Serial MSWF” implementation. To order to make the choice between the 2 types of MSWF implementation, simulations are carried out on various target group sizes starting from the value of 1 and extending to the full target size of 961. The results that are obtained will be discussed in the next section.

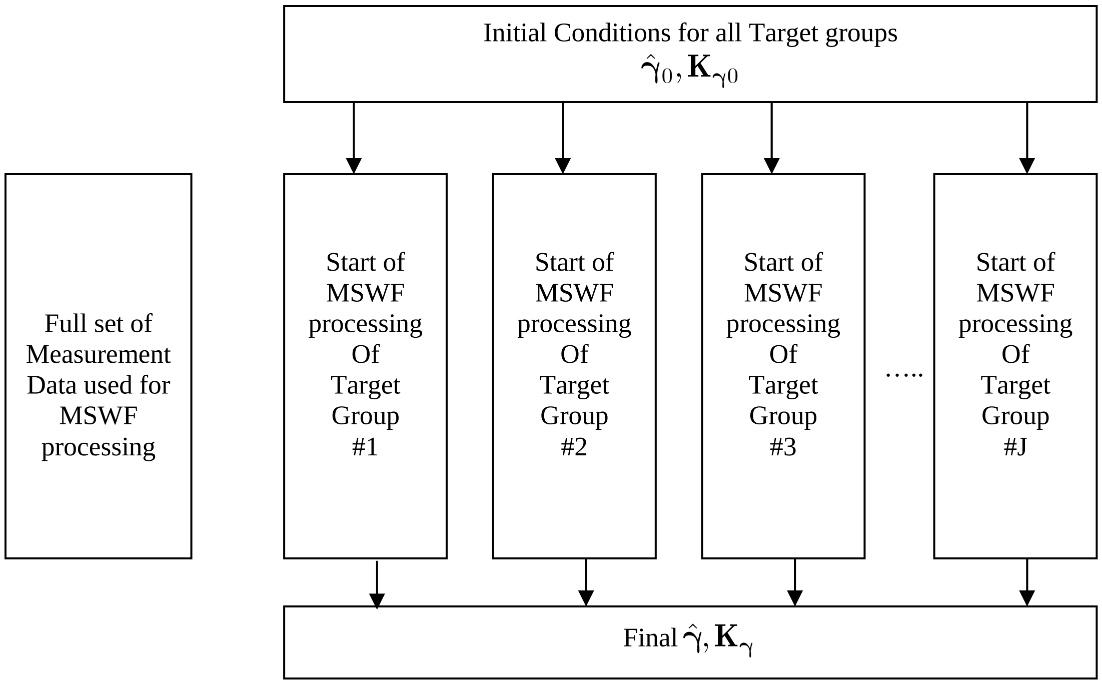
4.6 PARALLEL/SERIAL IMPLEMENTATION OF VECTOR MSWF

As mentioned in the previous section, the availability of vector MSWF capability allows for 2 types of implementation when processing all the targets or image resolution cells. To visualize the difference between these 2 types of vector MSWF implementation with that of the Kalman filter, some diagrams showing the structures of the Kalman filter, the parallel and serial implementation of the MSWF are provided. Note that in the following diagrams, $\hat{\gamma}_0 = 0$ and $\mathbf{K}_{\gamma 0} = \sigma_\gamma^2 \mathbf{I}$ as are defined in equation 2.27 in Chapter 2.



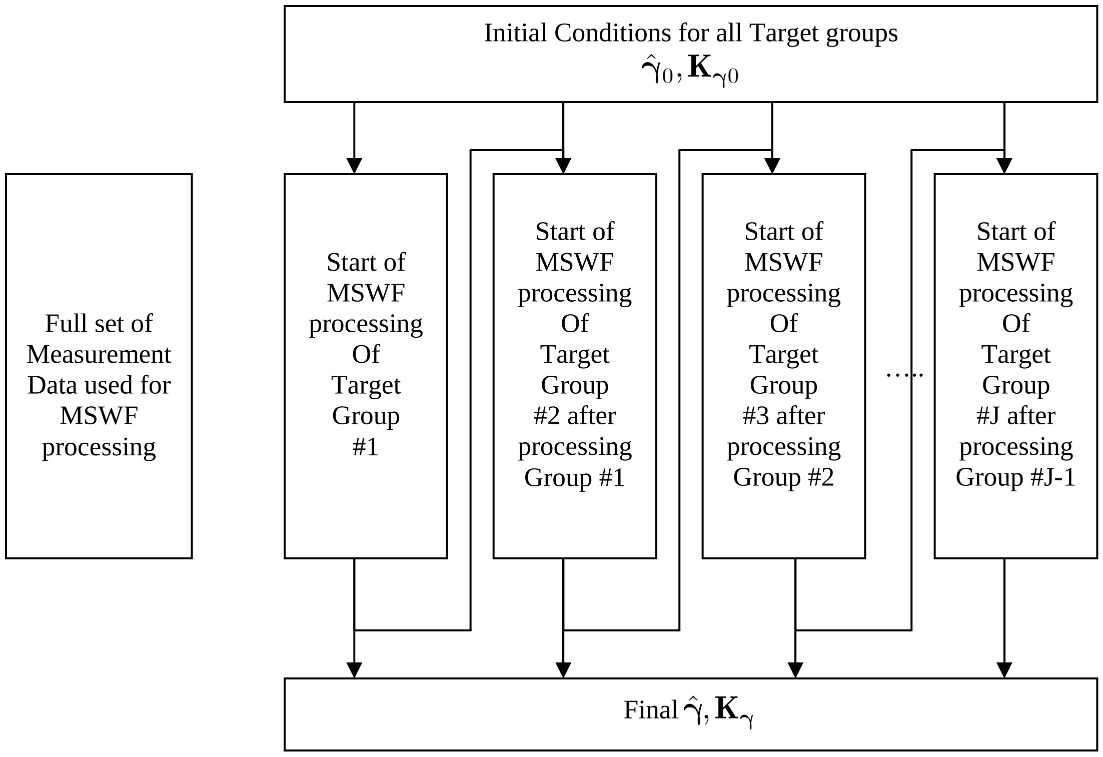
Initial Conditions
$\hat{\gamma}_0, \mathbf{K}_{\gamma 0}$
Measurement data subset 1
Simultaneous Processing of All Targets
Initial Conditions
$\hat{\gamma}_{11}, \mathbf{K}_{\gamma 11}$
Measurement data subset 2
Simultaneous Processing of All Targets
Initial Conditions
$\hat{\gamma}_{22}, \mathbf{K}_{\gamma 22}$
Measurement data subset 3
Simultaneous Processing of All Targets

Figure 4-5: Kalman Filter Implementation



Initial Conditions for all Target groups
$\hat{\gamma}_0, \mathbf{K}_{\gamma 0}$
Full set of Measurement Data used for MSWF processing
Start of MSWF processing Of Target Group #1
Start of MSWF processing Of Target Group #2
Start of MSWF processing Of Target Group #3
…..
Start of MSWF processing Of Target Group #J
Final $\hat{\gamma}, \mathbf{K}_{\gamma}$

Figure 4-6: Parallel Implementation of vector MSWF



Initial Conditions for all Target groups
$\hat{\gamma}_0, \mathbf{K}_{\gamma 0}$
Full set of Measurement Data used for MSWF processing
Start of MSWF processing Of Target Group #1
Start of MSWF processing Of Target Group #2 after processing Group #1
Start of MSWF processing Of Target Group #3 after processing Group #2
.....
Start of MSWF processing Of Target Group #J after processing Group #J-1
Final $\hat{\gamma}, \mathbf{K}_{\gamma}$

Figure 4-7: Serial Implementation of vector MSWF

From the above figures 4.5 to 4.7, we can notice the 2 basic differences between the Kalman filter and the vector MSWF. For Kalman filter, the full measurement data is divided into subsets for processing in each iteration step whereas in the vector MSWF, all measurements are used simultaneously. Also, in Kalman filter, all the targets are processed in each iteration step whereas in vector MSWF, the targets can be divided into different groups and processing is then performed independently on each group itself. Now, after having visualized the structures of the parallel and serial vector MSWF implementations, the results in terms of average expected error variance as defined in equation 4.24 and the total computation time needed with respect to target group sizes and stages of decomposition are then obtained and discussed in the following sections.

4.6.1 AVERAGE $\bar{\sigma}^2_{\varepsilon_0}$ VERSUS GROUP SIZE AND NUMBER OF STAGES

As mentioned in section 4.5, there is a need to decide on the size of the target groups to be used in the vector MSWF implementations. As such, one of the areas to be investigated is the relationship between the average $\bar{\sigma}^2_{\varepsilon_0}$ of all the targets versus the target group sizes and the stages of decomposition that are performed. Some plots of the relationship obtained are as shown below:

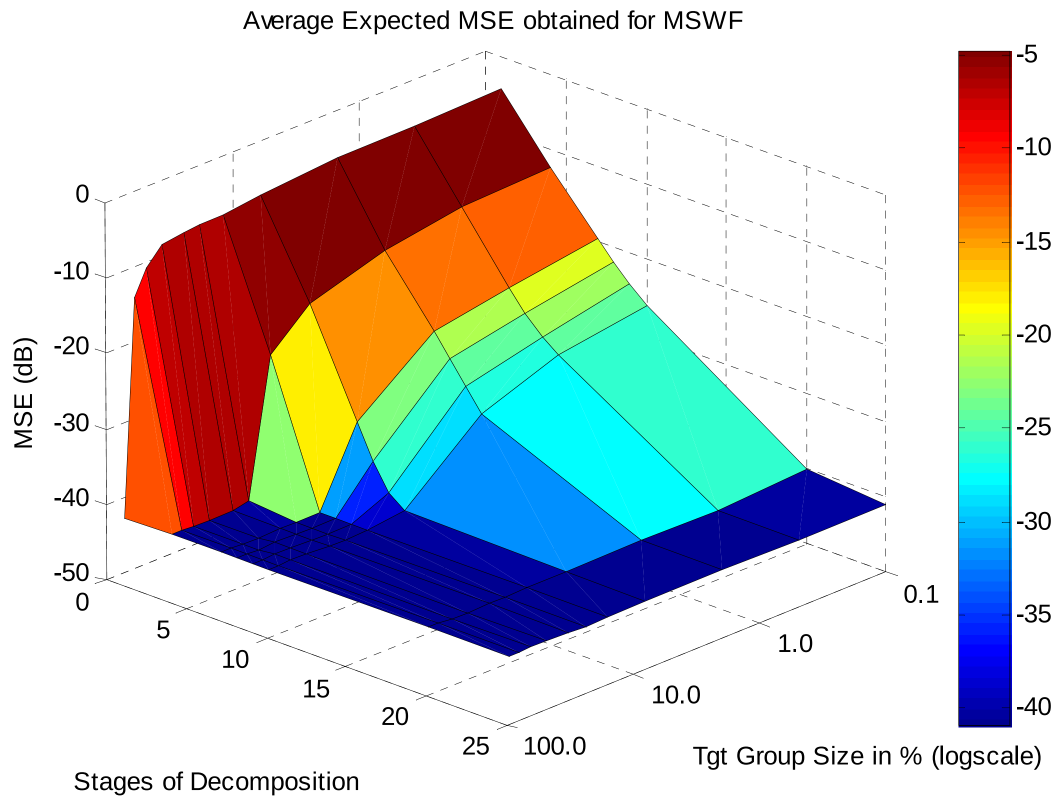


Figure 4-8: 3-D plot of Average MSE versus Group Size and Decomposition Stages

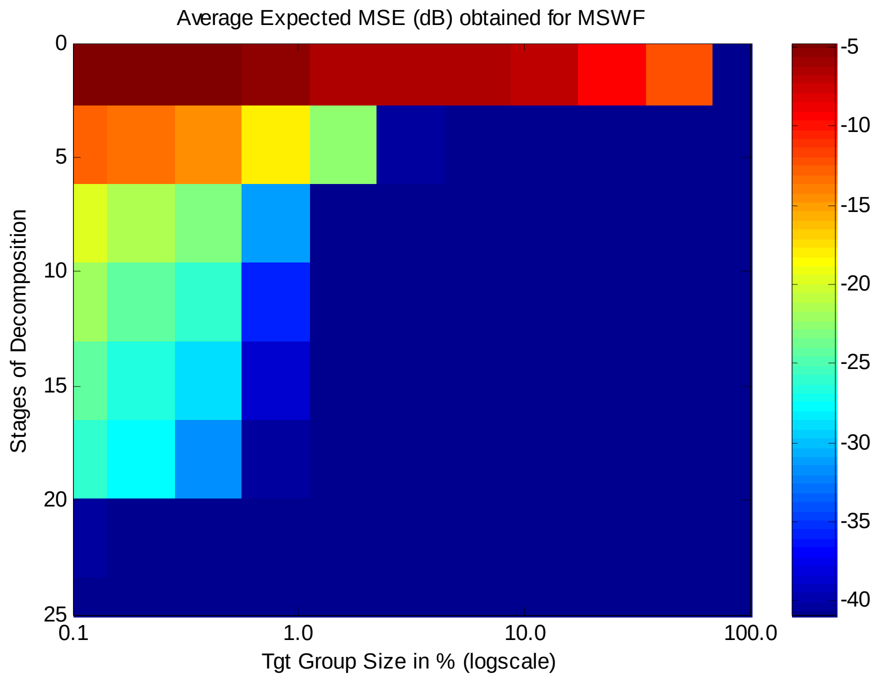


Figure 4-9: 2-D view of Average MSE for vector MSWF implementation

From figure 4.8, it can be seen again that as the size of the target group increases, it will require fewer stages of decomposition for the estimated values to obtain their steady state conditions. Furthermore, as the size of the target group approaches to 100% of the available targets, there is a sharp decrease in the number of decomposition stages required as seen by the steep slope in the figure 4.8. At the same time, we can also observed that there are many combinations in which steady state conditions can be reached as seen by the flat plane in the same figure itself. Finally, by examining figure 4.9, we can see again that when all the targets are used simultaneously in the decomposition, only 1 stage is required for convergence or steady state condition to be achieved as seen in the top right corner of figure 4.9.

4.6.2 TIME (PARALEL) VERSUS GROUP SIZE AND NUMBER OF STAGES

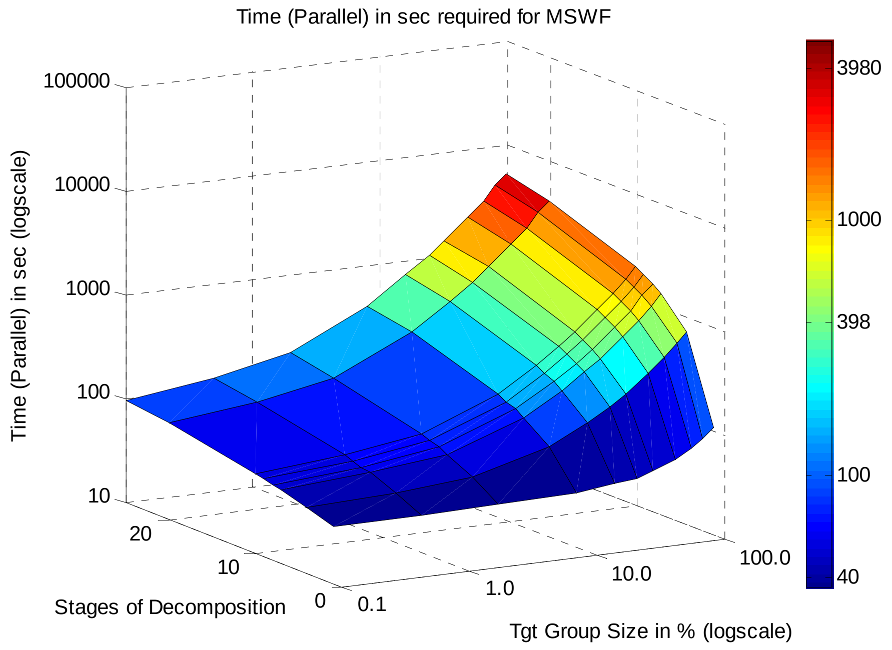


Figure 4-10: 3-D plot of Time for Parallel MSWF Implementation

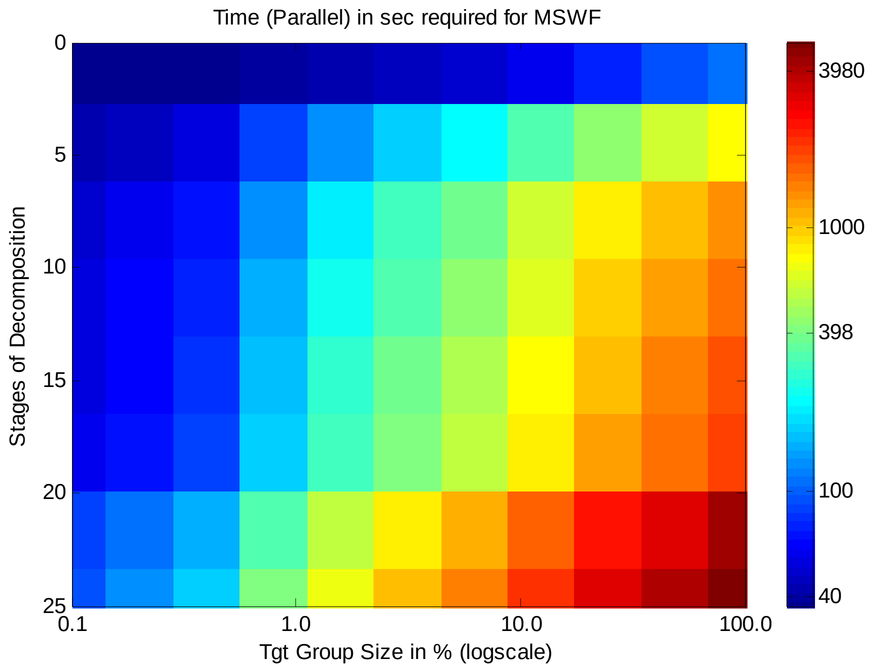


Figure 4-11: 2-D view of Time for Parallel MSWF Implementation

From figure 4.10 and figure 4.11, we are able to observe the expected trend that for parallel MSWF implementation, the computational time will increase exponentially when either the target group size or stages of decomposition or both increase in their values. At the same time, it is noted that the time difference between the smallest computational time and the largest computational time is of 2 orders of magnitude in difference.

4.6.3 TIME (SERIAL) VERSUS GROUP SIZE AND NUMBER OF STAGES

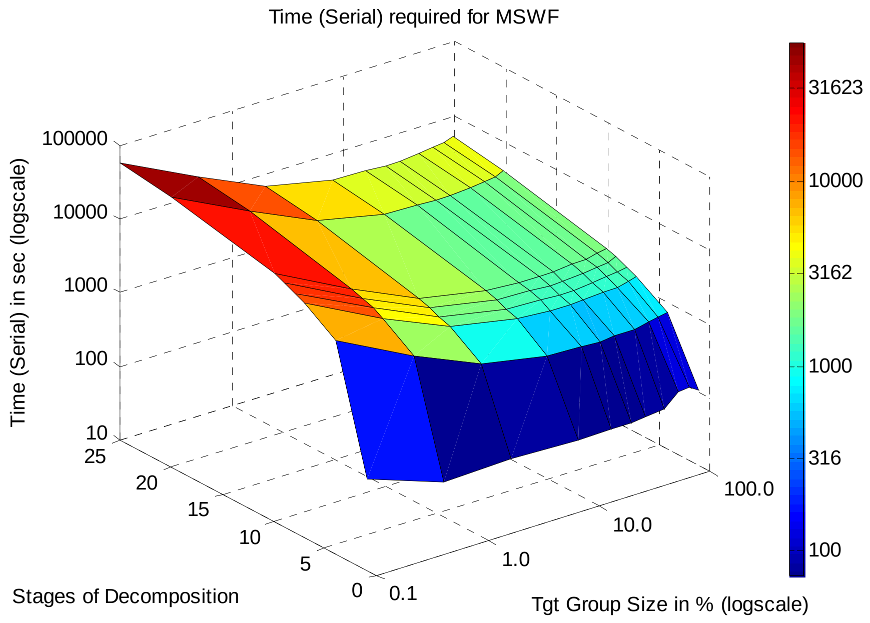


Figure 4-12: 3-D plot of Time for Serial MSWF Implementation

From figure 4.12, we can see that for serial MSWF implementation, as the target group size increases, the computational time required decreases as opposed to that of parallel MSWF implementation. However, as the target group size reaches 100%, the slope stops decreasing but climbs up instead. The main reason is because as there are more targets within the group, it will dramatically increase the time required to perform the non linear matrix inverse operation needed to obtain the normalized cross correlation matrix as shown in equation 4.18 and 4.19. The only exception seems to be at one location that is when the combination of all targets and 1 stage of decomposition is chosen as seen in the top right corner of figure 4.13.

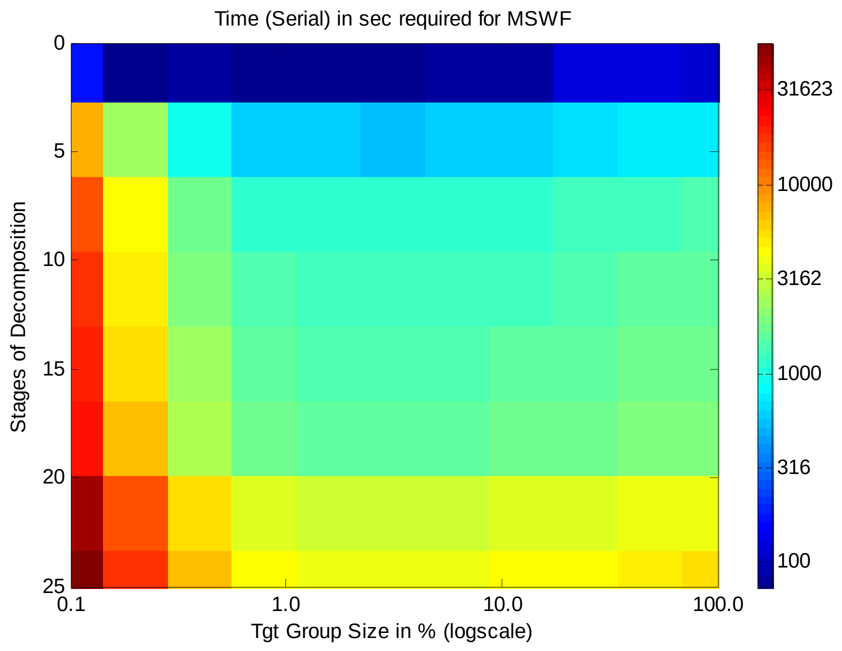


Figure 4-13: 2-D view of Time for Serial MSWF Implementation

4.6.4 CHOICE IN PARALLEL/SERIAL IMPLEMENTATION BASED ON MSE

After analyzing all the information provided in the previous 3 sections, the important decisions to make are to choose between parallel or serial MSWF implementation as well as the size of the target group itself. In order to make a good judgment between the various choices, I will place all the 2-D plots together so as to allow for better visualizations of the overall picture.

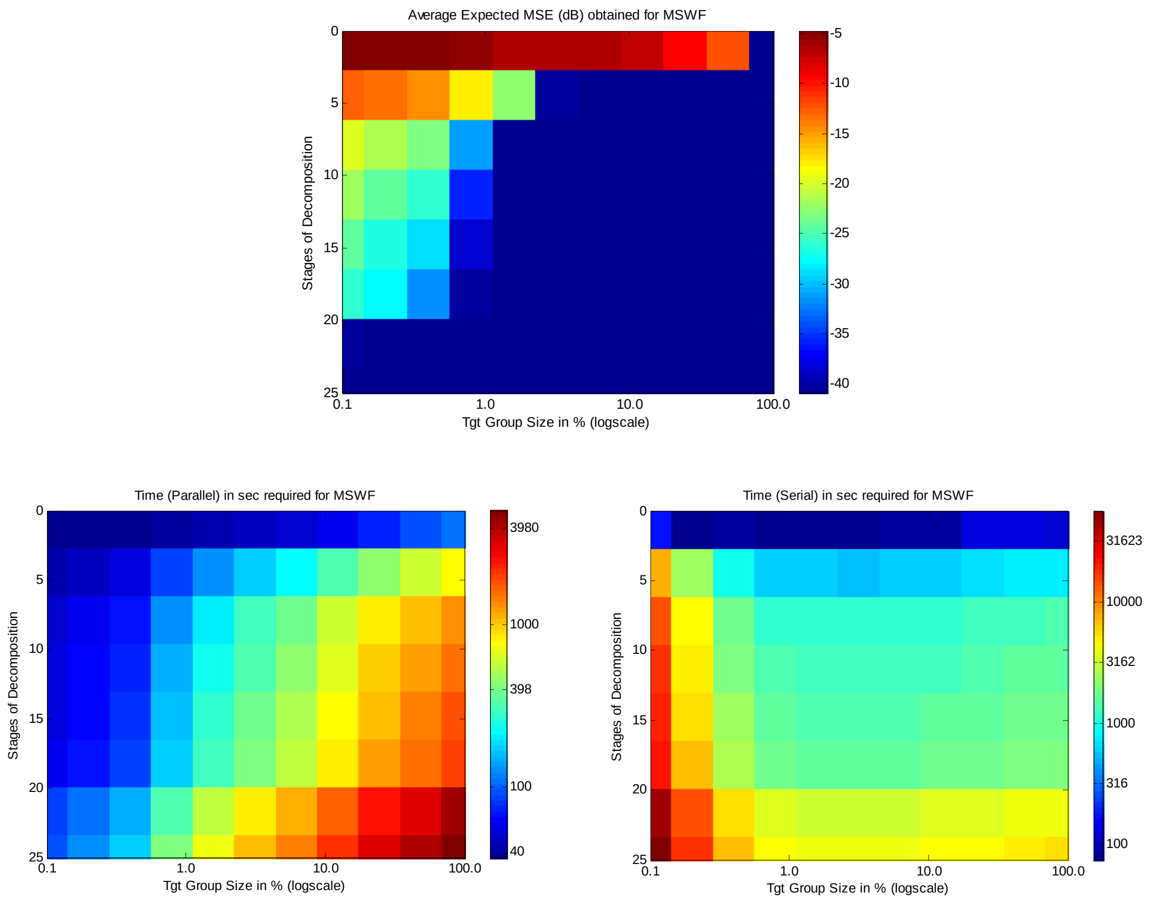


Figure 4-14: Relationship between Average MSE and Computation Time

By looking at figure 4.14, we can observe that in the region where there is convergence in the average expected MSE, it is generally better to use the parallel implementation approach as the computational time required is generally shorter. However, if there is a real constraint in the computing resources, using serial implementation will only increase the computation time by less than an order of magnitude in most cases which is acceptable. Also, for both parallel and serial implementations, there seems to be a common optimal combination at which convergence can occur rapidly. This combination is the choice of using all targets and using just 1 stage of the decomposition. However, it is also believed that as the number of targets increase dramatically from 961 to many thousands, this optimal condition may not hold true as ultimately, the matrix inverse operation required for computing the cross correlation matrix will become the dominant factor.

At this point, I have successfully completed the derivation as well as provided the implementation aspects and results of using the vector MSWF for solving the SAR image formation problem of a Non-Uniformly Distributed Multiple Aperture Radar system. But as the computational timing needed for convergence of the estimated values is rather high (112.572 seconds versus 35.262 seconds) as compared to the Kalman filter, one may wonder whether the Kalman filter still outperforms the MSWF when both the measurement data and target size increases. At the same time, it is also curious to know whether the MSWF will still hold its edge over the MMSE filter (112.572 seconds versus 186.078 seconds) in such a situation. In order to answer this question, a simulation is performed on the 3 types of filter again but using a much larger data set. The results of this simulation are shown in the next section.

4.7 RUNNING LARGER DATASET FOR PARALLEL MSWF

In this section, a simulation is performed using the MMSE filter, the Kalman filter as well as the parallel vector MSWF on a data set that is about 4 times as large when compared to the simulations that are run previously. As the computing resources needed for this simulation is beyond that of a standard PC computer, it is therefore executed in the freestyle server machine in the premise. Note that as the freestyle server is not a dedicated machine for performing this simulation but accessed by many users, thus many simulation runs have to be performed for each scenario before the average results are obtained. In Table 4.11 below, the parameters that are used for the simulation are as shown:

Table 4.11: Parameter Values used for Large Data Set Simulation

S/N	Description of Parameters	Values chosen
1	N_x	63
2	N_y	63
3	Full Filter Rank size (= $N_x \times N_y$)	3969
4	Total number of transmitters	1
5	Total number of receivers	12
6	Total number of samples	11460
7	SNR (Signal to Noise Ratio)	40 dB

Using the parameters shown in the above Table 4.11, the simulation is performed on all 3 filters with an measurement group size of 191 measurements per iteration step being used for the Kalman filter. This value of 191 is only determined after many runs are performed on the Kalman filter to ensure that the measurement size of 191 per iteration step will provide an optimal computational time using the Kalman filter. As for the MSWF, it is carried out by grouping all the targets in a single group and using 1 stage of decomposition to obtain the final results. Next, the results obtained for the simulation are then recorded and shown in Table 4.12 on the next page.

Table 4.12: Large Data Set Results from MMSE, Kalman filter and MSWF

Filter Type	Total Time /sec	Average $\overline{\sigma}_{\varepsilon_0}^2$ /dB	Average Computed MSE /dB
Wiener	9349.3	-39.483	-39.439
Kalman	1192.1	-39.483	-39.439
MSWF	3432.5	-39.479	-39.439

From the results shown in Table 4.12, we can conclude that the MSWF will generally not be more computational efficient as compared to the Kalman filter. However, the MSWF edge over the MMSE or Wiener filter will improve tremendously when the measurement data size and target size increase significantly.

Finally, the next question that comes to mind will be to determine whether the vector MSWF timing decreases by using some innovative or pre-processing approaches in its implementation. This question will be answered in the next Chapter on using innovative implementation of the MSWF.

CHAPTER 5: INNOVATIVE MSWF IMPLEMENTATIONS

5.1 USING MODIFIED APPROACH TO INITIALIZATION OF DATA

In the previous Chapter, I have discussed both the scalar and vector implementation of the Multi-Stage Wiener filter (MSWF) along with using either parallel or serial computing architecture for the vector MSWF implementation. Although the MSWF functions properly using each of this combination, there lies a question on whether its performance improves in terms of the computational time required. In order to answer this question, one can firstly re-examine the structure of the efficient Kalman filter as shown in figure 4.5 and reproduced again in figure 5.1 below to forge some idea on speeding the processing of the MSWF.

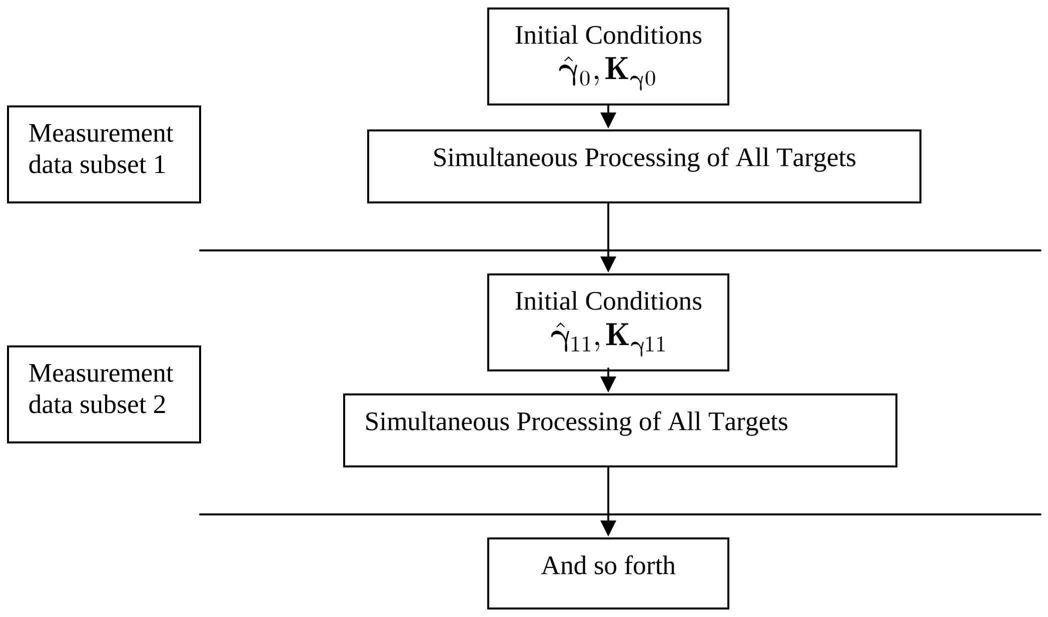

Initial Conditions
$\hat{\gamma}_0, \mathbf{K}_{\gamma 0}$
Measurement data subset 1
Simultaneous Processing of All Targets
Initial Conditions
$\hat{\gamma}_{11}, \mathbf{K}_{\gamma 11}$
Measurement data subset 2
Simultaneous Processing of All Targets
And so forth

Figure 5-1: A re-look at the structure of Kalman Filter

By examining figure 5.1, we can notice that the initial conditions used for processing the next batch of measurement data input to the Kalman filter are not the same as the initial conditions used at the processing of the previous batch of measurement data. Instead, the new initial conditions $\hat{\gamma}_{11}, \mathbf{K}_{\gamma 11}$ are the output of all the targets from the Kalman filter processing of the 1st batch or subset of measurement data. By repeatedly refining the initial conditions that provide a closer picture to the actual values of the states to be estimated, the Kalman filter is thus able to achieve convergence once enough measurements are processed. Essentially, the Kalman filter's approach can be summarized as using piecewise measurement data processing along with refinement of initial conditions using output from previous processing.

Looking at this approach, one can draw some analogy between the Kalman filter structure and the serial implementation of the MSWF that is shown in figure 4.7 in Chapter 4. For instance, if the initial conditions for processing the 2nd group of targets in the serial MSWF implementation are also refined using the output from the 1st group of targets, then the division is in the target data that is analogous to the division in the measurement data using the Kalman filter. However, unlike the Kalman filter where the $\hat{\gamma}_{11}$ vector consists of the estimates of all targets from the previous batch of data, the $\hat{\gamma}_{1}$ vector used in the 2nd target group for the serial MSWF processing consists of a mixture of estimates from the 1st target group and the rest from the original initial condition $\hat{\gamma}_{0}$. Similarly, the matrix $\mathbf{K}_{\gamma 1}$ used in the initialization of the 2nd target group for the serial MSWF processing is not a fully filled matrix but rather a mixture of diagonal and block diagonal elements. This analogy can be more easily visualized in the modified serial MSWF implementation as shown in figure 5.2 below:

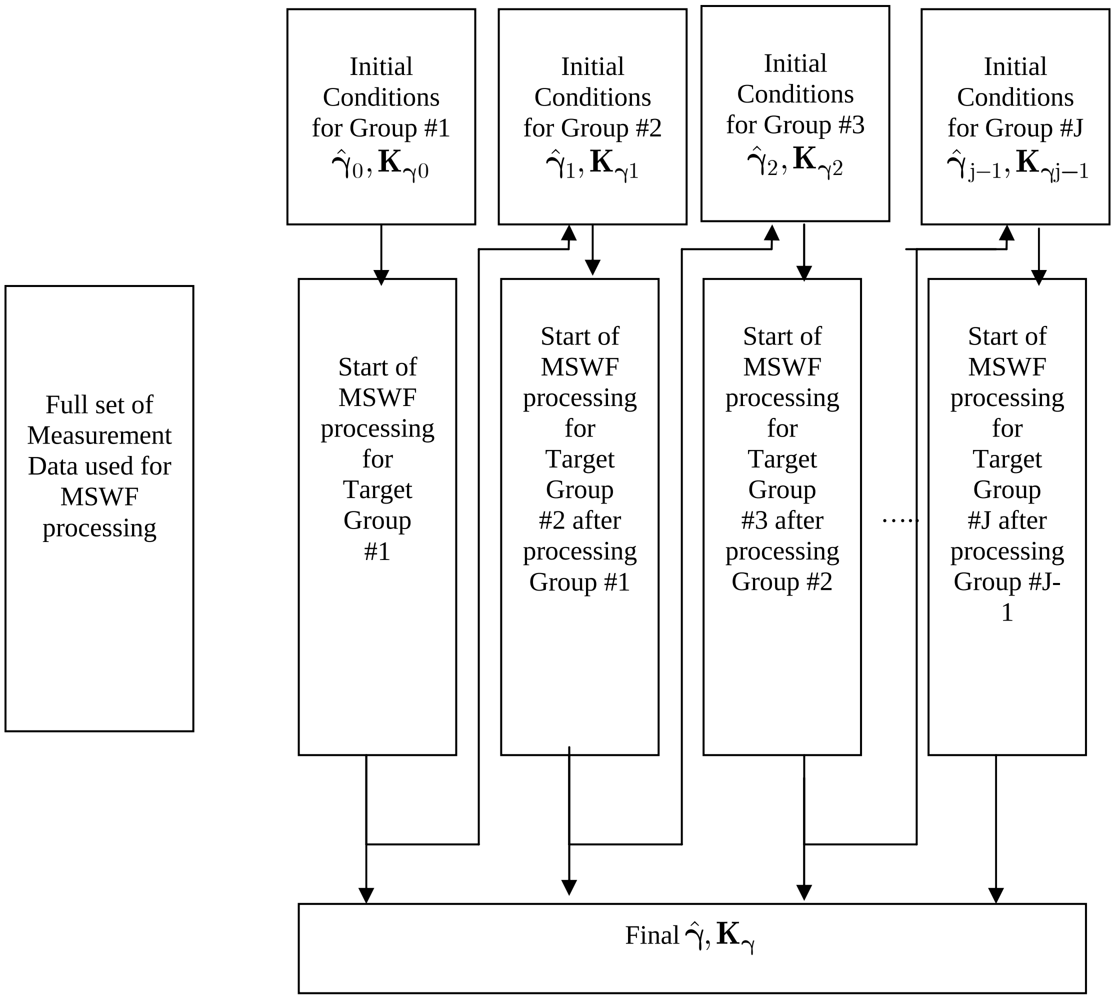

Initial Conditions for Group #1 $\hat{\gamma}_0, \mathbf{K}_{\gamma 0}$
Initial Conditions for Group #2 $\hat{\gamma}_1, \mathbf{K}_{\gamma 1}$
Initial Conditions for Group #3 $\hat{\gamma}_2, \mathbf{K}_{\gamma 2}$
Initial Conditions for Group #J $\hat{\gamma}_{j-1}, \mathbf{K}_{\gamma j-1}$
Full set of Measurement Data used for MSWF processing
Start of MSWF processing for Target Group #1
Start of MSWF processing for Target Group #2 after processing Group #1
Start of MSWF processing for Target Group #3 after processing Group #2
Start of MSWF processing for Target Group #J after processing Group #J-1
Final $\hat{\gamma}, \mathbf{K}_{\gamma}$

Figure 5-2: Modified Serial Implementation analogous to Kalman filter

Using this new approach, a simulation is performed on the modified serial MSWF implementation approach using 3 different target group sizes of 20, 100 and 480 so as to analyze the trend of the end results obtained from the MSWF using this approach. The results are shown in figure 5.3 in the following page.

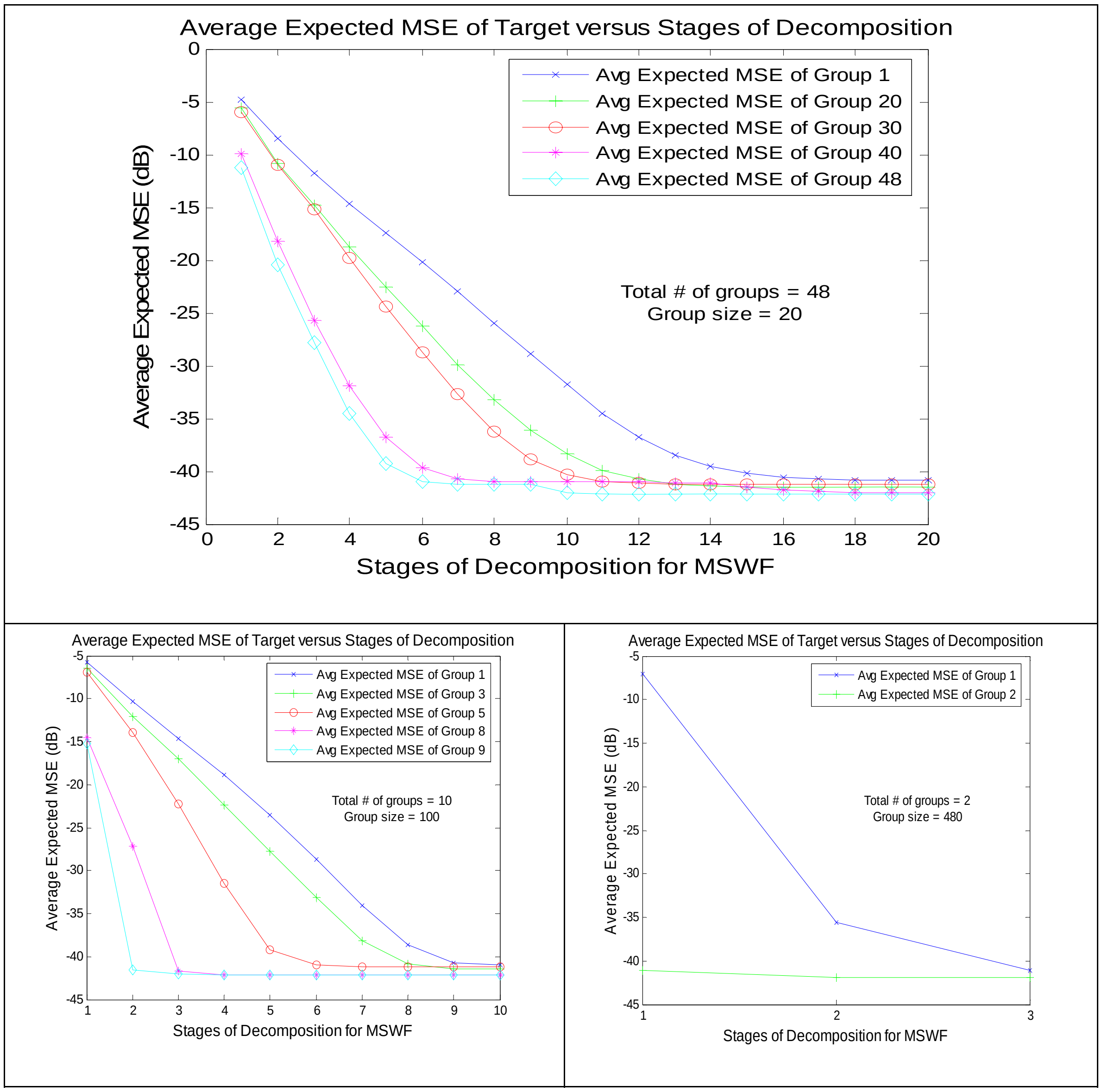


Figure 5-3: Results using Modified Serial MSWF Implementation

In figure 5.3 above, the rate of convergence of each target group with respect to stages of decomposition is plotted for various target group sizes of 20, 100 or 480 targets per group. To start, let's examine the topmost plot in which there are 20 targets per target group. Comparing the rate of convergence between the 1st target group and the 20th target group, we can see that the 20th target group has a faster convergence rate as it only needs 12 stages of decomposition to reach its steady state values whereas the 1st target group needs at least 18 stages of decomposition to achieve steady state conditions. If we are to look at the convergence rate of the last target group or the 48th target group in the list, we will see that it has achieved its steady state condition in just 6 stages of decomposition. Next, if we are to continue examining the bottom 2 plots in which the target group size are namely 100 and 480, we can notice the similar trends in these plots as well. This improvement in the convergence rate is most prominent in the case where the target group size is 480, in which the 2nd target group is able to achieve convergence at just the 1st stage of decomposition whereas the 1st target group needs 3 stages to achieve convergence condition.

Next, after discovering that the rate of convergence for the target groups will improve when the modified serial MSWF implementation, the question will be to determine how this improvement can be linked to the computational time needed for processing all the targets. Looking at the 3 plots in figure 5.3 again, we can hypothesize that on average, the number of stages of decomposition required for each target group has been reduced by about 40% as compared to the situation when all target groups are initialized with the same initial conditions. For example, by looking at the plot where the target group size is 100, we can conclude that the average number of stages required for convergence is about 6 stages versus that of 10 stages of decomposition if the initial conditions are identical across all target groups. This should translate in layman terms to roughly about 40% improvement in the computational time needed. To support this hypothesis, the total computational time taken for all target groups to achieve convergence is measured for the original serial MSWF implementation as well as that of the modified serial MSWF implementation. The results are recorded and shown in Table 5.1 below:

Table 5.1: Timing results from 2 types of Serial MSWF Implementation

Filter Type	**Total Time /sec**
Serial MSWF	1758.0
Modified Serial MSWF	1278.3

Looking at the 2 timing results obtained, we can see that the modified serial MSWF implementation has indeed improve the performance by a factor of about 28%. The main reason that the improvement is not as good as the 40% predicted by the earlier hypothesis is because unlike the standard serial MSWF implementation where the initial condition matrix $\mathbf{R}_{\mathbf{x}_0}$ (refer to equation 4.10) is prepared just once for all target groups, there is now a need to re-compute this $\mathbf{R}_{\mathbf{x}_0}$ for each target group since the matrix $\mathbf{K}_\gamma$ used to compute $\mathbf{R}_{\mathbf{x}_0}$ is now different for different target groups. As a result, this new requirement becomes an overhead to the computational cost and reduces the overall time savings to 28% instead of 40% as predicted in the hypothesis. Nevertheless, achieving a saving of 28% in the total computational time is very significant and thus the modified serial MSWF implementation should be preferred in all situations in which serial implementation is chosen over parallel implementation.

Next, after having successfully introduced some improvement to the serial MSWF implementation, the subsequent step will be to look at alternative approaches to the parallel MSWF implementation approach that may allow its performance in terms of computational time to be improved. The details will be discussed in the next section.

5.2 USING MODIFIED TARGET GROUPING APPROACH

Now, in all the previous simulations and results obtained for the MSWF, the targets are grouped based on their spatial proximity in the regular target grid area. For example, target 2 is just located in between target 1 and target 3 in the whole target area itself. Although there is nothing wrong with this form of grouping scheme, the question arises as to whether another form of grouping scheme will make any difference to the performance of the MSWF. To answer this question, we must examine the inherent mechanism of the MSWF itself. Now, from Chapter 4, we understand that the basic feature of the MSWF is to work on removing the residual correlations between targets (like a whitening process) such that each target's final output will be free of correlation effects from other targets. Thus, using some insight or logical deduction, one would hypothesize that the target of interest will be able to achieve convergence at a faster rate if targets that are most correlated to it are placed in the same target group as it is such that the effects of their correlation are quickly removed from the target's output. Using the same line of thinking, one can also hypothesize that the target of interest will achieve convergence at a slower rate if targets that are least correlated to it are grouped together with it. Therefore, to test these two hypotheses, two grouping schemes are devised so as to allow targets to be grouped together based on their least or highest correlation to at least 1 target of interest per group. For ease of tracking, I will call the 1st grouping scheme “scheme A” and the 2nd grouping scheme “scheme B”. In the following sections, I will discuss on the details of each grouping scheme as well as the simulation results obtained after using these 2 schemes on the parallel MSWF implementation.

5.2.1 SCHEME A – GROUP TARGETS BASED ON LEAST CORRELATION

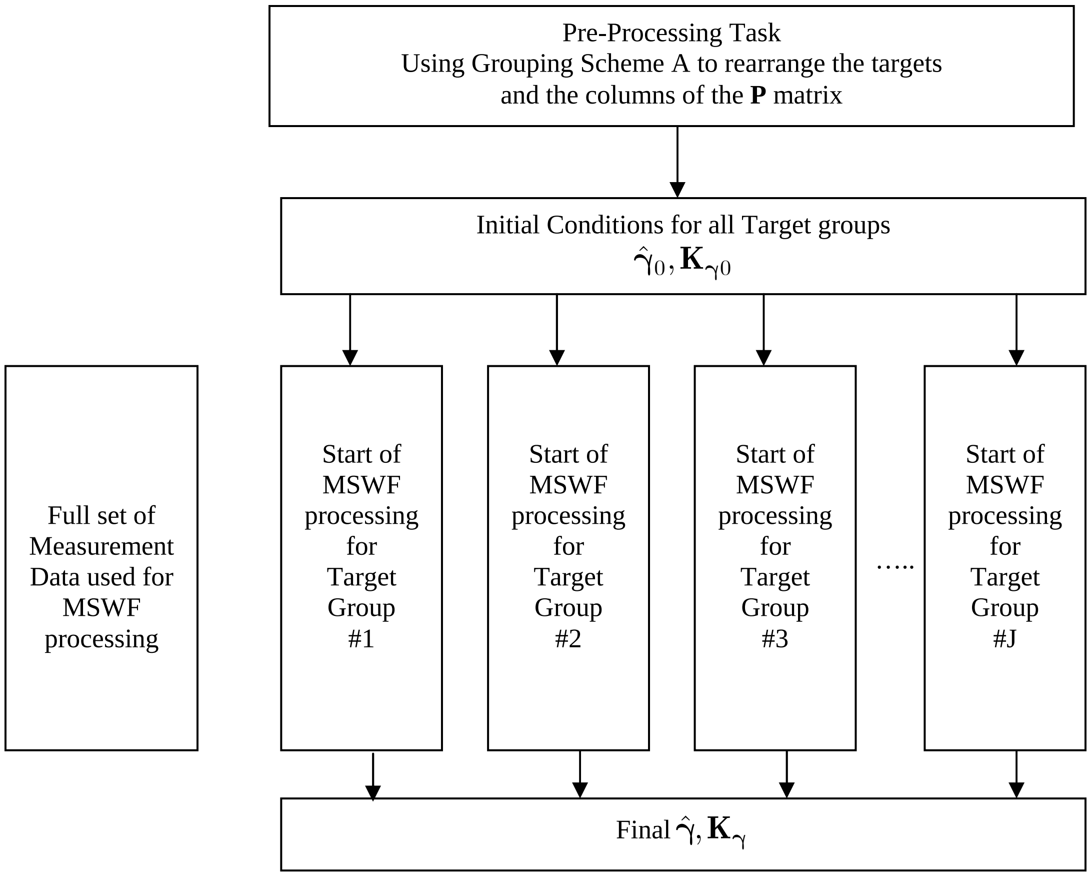
In this scheme itself, a computation is first performed on the **P** matrix so as to obtain the cross correlation magnitude between targets. The computation is as follows:

$$\begin{aligned} \mathbf{P}_1 &= \mathbf{P}^{\mathrm{H}}\mathbf{P} \\ \mathbf{P}_2 &= |\mathbf{P}_1| \end{aligned} \qquad (5.1)$$

In equation 5.1, $\mathbf{P}_2$ will obtain the magnitude of the auto correlation and cross correlation magnitudes of all the targets in each of its element. Also, the size of matrix $\mathbf{P}_2$ will be equal to $N_t * N_t$ where N_t is equal to the total number of targets. Next, within each column of $\mathbf{P}_2$, a sorting operation is performed on the elements in the column such that the smallest value will be at the top of the column and the largest value (largest value will be equal to the auto correlation magnitude) will be at the last element of the column, i.e. in ascending order. This sorting operation is then performed on all the columns inside $\mathbf{P}_2$. Once this is done, the next step will be to start the grouping of the targets.

In the grouping operation, the required size of the target group K is first determined, for example 20, 40 or 100 targets per group. Secondly, target 1 of the original target numbering is chosen to be also the 1st target of the new grouping scheme. Now, target 1 will be located at the last element of column 1 in $\mathbf{P}_2$ after sorting since column 1 contains both the auto correlation magnitude $\mathrm{E}\{\boldsymbol{\rho}_1^*\boldsymbol{\rho}_1\}$ of target 1 and the cross correlation magnitudes $\mathrm{E}\{\boldsymbol{\rho}_1^*\boldsymbol{\rho}_{j,j\neq 1}\}$ of other targets to target 1 and $\mathrm{E}\{\boldsymbol{\rho}_1^*\boldsymbol{\rho}_1\}$ is the largest magnitude in that column. The grouping scheme begins by taking the K-1 targets starting from the 1st element in column 1 that are least correlated to target 1. These K targets will then form the new target group #1 with target numbering from 1 to K. Next, the focus shift to column 2 and the original target 2 in the 2nd column is checked to ensure that it has not yet been included as one of the K-1 targets in the new target group #1. If so, then the focus will shift to column 3 of the $\mathbf{P}_2$ instead. If not, then target 2 will become the 1st target of the 2nd target group and K-1 targets starting from the 1st element in column 2 will be added to target 2 to form the new target group #2. Note that the K-1 targets are chosen such that they are distinct from those targets already included in the previous new target grouping. Also, the original target 2 will become the (K+1)th target in the new target numbering process. This grouping operation is then performed repeatedly until finally all N_t targets have been grouped into the new target groupings. Finally, using the new target numbers, the columns of the **P** matrix are also rearranged based on the new target number ordering.

To test the 2nd hypothesis that is mentioned in section 5.2, the newly grouped targets then undergo the parallel MSWF implementation and the expected *MSE* of the 1st target in each new target group is then compared with its original expected *MSE* using the old grouping scheme for each stage of decomposition. A diagram showing the modified approach to the parallel MSWF implementation is as shown in figure 5.4 below:



Pre-Processing Task
Using Grouping Scheme A to rearrange the targets
and the columns of the **P** matrix
Initial Conditions for all Target groups
$\hat{\gamma}_0, \mathbf{K}_{\gamma 0}$
Full set of Measurement Data used for MSWF processing
Start of MSWF processing for Target Group #1
Start of MSWF processing for Target Group #2
Start of MSWF processing for Target Group #3
…..
Start of MSWF processing for Target Group #J
Final $\hat{\gamma}, \mathbf{K}_{\gamma}$

Figure 5-4: Modified Parallel Implementation of MSWF

The simulation again uses 3 different target group sizes of 20, 100 and 480 as in section 5.1 and the results obtained are shown in figure 5.5 below:

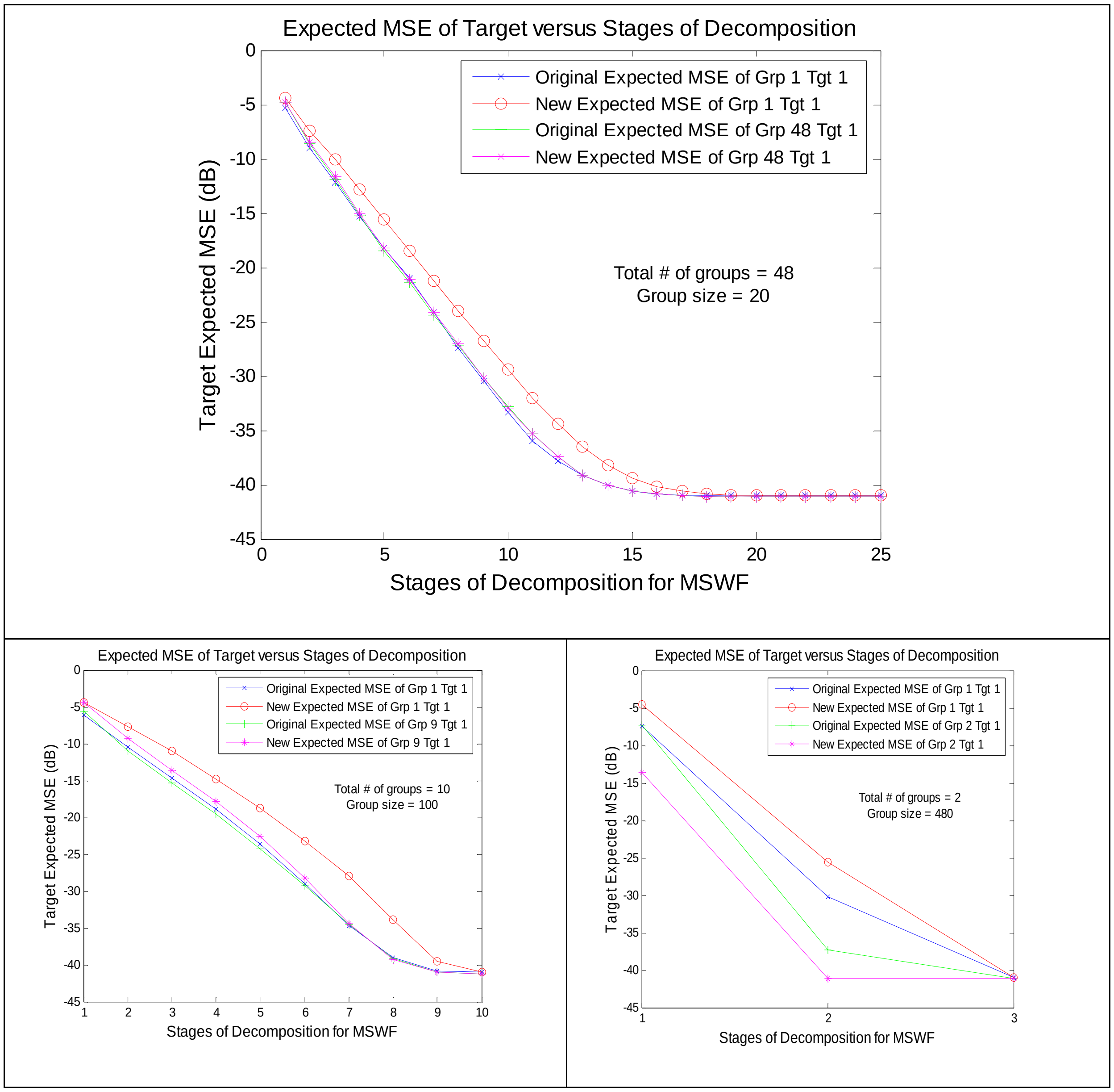


Figure 5-5: Results using Parallel MSWF Implementation (Least correlation)

Looking at the topmost plot in figure 5.5 where the target group size is 20, we can observe that the new rate of convergence for target 1 is slower than the original rate of convergence when grouping scheme A is used. However, at the last of the target groups, i.e. group #48, the impact of the new grouping scheme has diminished such that there is hardly any difference in the new convergence rate and the old convergence rate. Next, by examining the plot where the target group size is 100, we can observe the same trend as is shown in the previous plot. Thus, from the results of these two plots, we can say that the 2nd hypothesis in section 5.2 which says that the target of interest will achieve convergence at a slower rate if targets that are least correlated to it are grouped together with it is true to a certain extent. As for the reason that there is hardly any difference in the convergence rate of the last target group, the reason is because the grouping scheme A is biased in that as more and more distinct targets are chosen for the earlier target groupings, the *K*-1 remaining targets to be placed in the last target group may or may not be the least correlated to the 1st target of the last target group. Thus, the behavior is atypical from the earlier groups.

Next, coming to the 3rd plot of figure 5.5 in which the size of the target group is 480, we notice one interesting phenomenon. For the new convergence rate in the 2nd target group of this plot, instead of having the same behavior as that of the other two plots, instead it is having a faster convergence rate as compared to the original convergence rate in the 2nd target group. The main reason is again due to the bias that is present in the grouping scheme A. Since so many targets that are least correlated to the 1st target in target group #1 have been included in the target group #1 itself, therefore the remaining targets in target group #2 are more likely to be highly correlated to the 1st target in the same group which resulted in a faster, not slower convergence rate.

At this point of time, after verifying the 2nd hypothesis defined in section 5.2 to a large extent, the next task will be to try to verify the 1st hypothesis in section 5.2 which states that the target of interest will be able to achieve convergence at a faster rate if targets that are most correlated to it are placed in the same target group as it is. The details of the grouping scheme B for verifying this hypothesis and the results are discussed in the following section.

5.2.2 SCHEME B – GROUP TARGETS BASED ON HIGHEST CORRELATION

In scheme B target grouping, the steps and processes are very similar to that of scheme B target grouping with only 2 exceptions. Firstly, the elements within each column of $\mathbf{P}_2$ are sorted in a descending order manner rather than in ascending order. Secondly, the grouping scheme B will begin by taking the K-1 targets starting from the 2nd element in column 1 that are most correlated or highly correlated to target 1 (target 1 is now the 1st element in column 1 due to the sorting in descending order). These K targets will then form the new target group #1 with target numbering from 1 to K. Similarly, the column operations continue until finally all N_t targets have been grouped into the new target groupings using scheme B.

Next, to test the 1st hypothesis defined in section 5.2, these newly grouped targets also undergo the parallel MSWF implementation and the expected *MSE* of the 1st target in each new target group is then compared with its original expected *MSE* for each stage of decomposition. The simulation also uses the same target group sizes of 20, 100 and 480 as in section 5.2.1 and the results obtained are shown in figure 5.6 below:

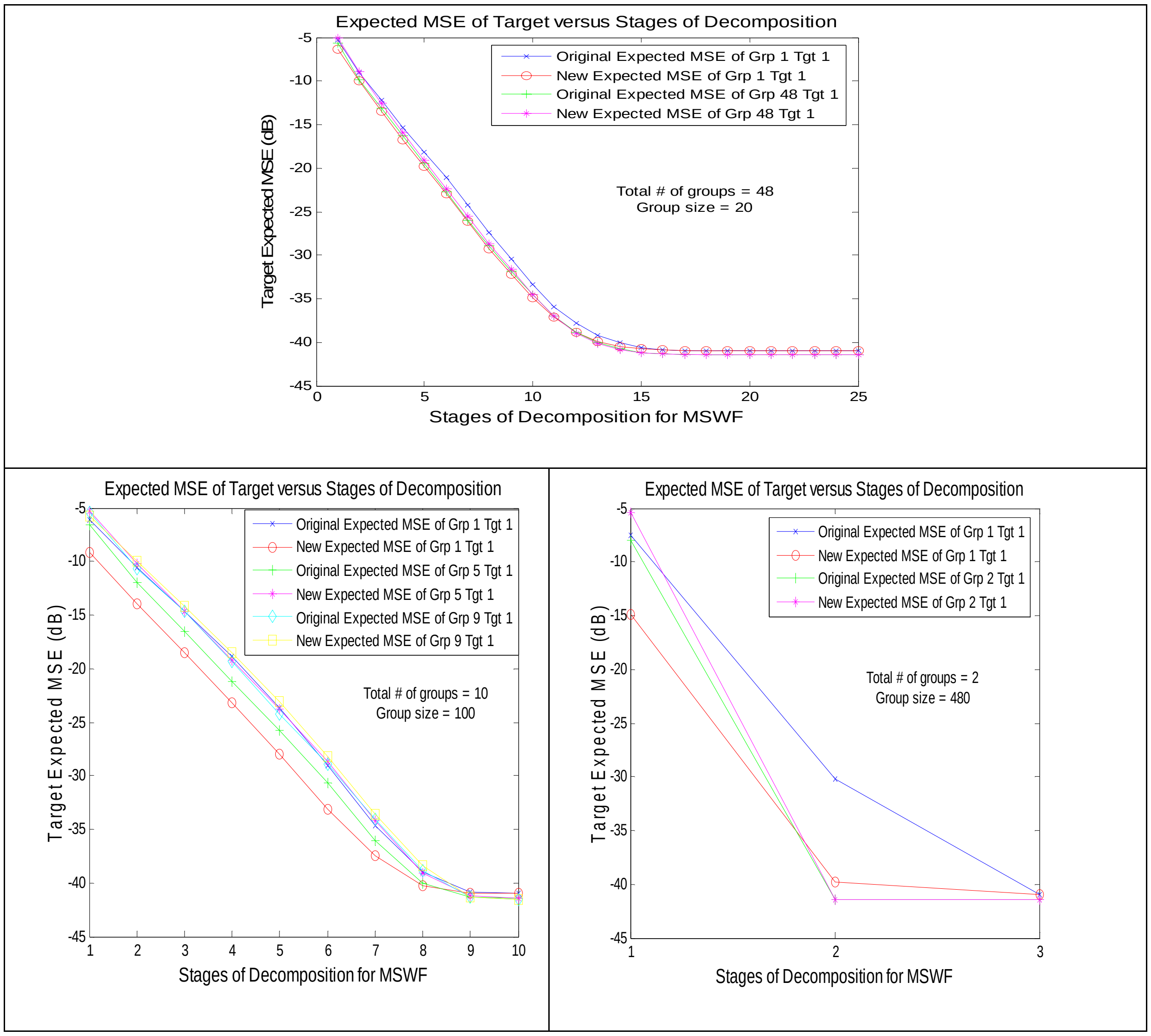


Figure 5-6: Results using Parallel MSWF Implementation (Highest correlation)

Now, by looking at the topmost plot in this figure, we can see that the new rate of convergence for the 1st target in target group #1 has indeed improved over its original rate of convergence after grouping scheme B is applied. Thus, this result is indicating that the 1st hypothesis may be true. Next, as was shown in grouping scheme A in section 5.2.1, there is hardly any difference between the new convergence rate and the original convergence rate of the 1st target in the last target group, i.e. target group #48. This is due to the same fact that as grouping scheme B is also biased in that as most of the distinct targets are chosen for the earlier target groupings, the K-1 remaining targets to be placed in the last target group may not be correlated to the 1st target of the last target group at all but rather they are just the leftover targets. As such, the trend in the earlier target groups is not seen in the latter target groups and especially in target group #48, which is the last target group.

Next, by examining the plot with target group size equal to 100 at the bottom right corner of figure 5.6, we can observe some surprising trends. Although we expect to see that the new convergence rate for the 1st target in target group #1 to be faster than its original convergence rate using grouping scheme B in support of hypothesis 1, we never expect at all to see the reverse trend for the 1st target in the 5th target group. In fact, after examining the collected results closely, it is found that the trend has actually started reversing itself at the 4th target group and the trend then swings back and forth between the 6th target group and the last target group. After careful analysis of the elements within the matrix $\mathbf{P}_2$ so as to explain this surprising trend, it is discovered there are not so many targets that are correlated to the target of interest in each column, i.e. the number of significant cross correlation values per column is much lesser than 100. Thus, by using a target group size of 100 (100 is too large) for the not so perfect scheme B, it has resulted in prematurely taking away targets that are actually correlated to the target of interest in the latter target groups. Thus this defeat the original purpose of grouping targets together based on high correlation. In a sense, this means that grouping scheme B has failed in this situation but the 1st hypothesis still remains true. This revelation is supported by the plot for the target group size of 480 in which the new convergence rate for the 1st target in the 1st target group is faster than its original convergence rate. However, the new convergence rate for the 1st target in the 2nd target group is slower than its original convergence rate at the early stages of decomposition before changing to about the same value towards the end.

Thus, at this stage, we have managed to verify both hypothesis 1 and 2 defined in section 5.2 that states that grouping of targets based on high cross correlation should help in increasing the rate of convergence when using the MSWF and the reverse will also hold true when the targets that are very least correlated are grouped together instead. However, due to the bias or defects of both grouping scheme A and B, these schemes are not able to fulfill their tasks completely such that the latter target groups are not grouped according to the desired criterion. As a result, I am not able to show via simulation results that there is an actual improvement in the computational time needed when using parallel MSWF implementation in conjunction with scheme B as the net convergence rate across all target groups remains unchanged when applying this scheme. Nevertheless, I believe that if a more intelligent grouping scheme is designed such that it allows the targets that are highly correlated to each other to be grouped together without any preference given to the earlier target groupings, this approach will definitely improve the performance of the MSWF in either the parallel and serial implementation methods.

Also, at this point in time, we have already looked at 3 approaches that involve either the targets' initialization conditions or their style of grouping that may help to alter the performance of the MSWF in terms of timing requirement. Basically, these approaches deal mainly with the target space. Looking ahead, the next approach that I will be attempting will involve the measurement space as well. By drawing inspiration from Kalman filter that is able to execute much faster than the MMSE or Wiener filter because it breaks up the total measurement data set into subsets in its processing, the question for MSWF is whether can its processing be sped up likewise. Essentially, this implies that if the measurement data set is broken into various subsets in the MSWF implementation, it may also help to reduce its computational time as well. Thus, to answer this question, the original structure of the MSWF implementation as shown in Chapter 4 is modified again and the results obtained are discussed in the next section.

5.3 USING MEASUREMENT SUBSETS FOR RECURSIVE MSWF

In Kalman filter, the processing is done recursively or iteratively on every new subset of measurement data that it has just received. As such, the Kalman filter can also be known as the recursive or iterative MMSE filter. Now, if it is possible for the MSWF to process new incoming measurement data in an iterative manner without having to re-process all the previous data again, it can also be known as the recursive or iterative MSWF.

Next, to allow the MSWF to be implemented in a recursive manner, some modifications will have to be made to the initial conditions of the recursive MSWF at the 2nd iteration step as compared to the first iteration step. Essentially, the concept of innovation data used in the Kalman filter processing will also be required in this situation starting from the 2nd iteration step of the recursive MSWF. Besides that, other issues like updating the new estimates of the scattering coefficient as well as the Error covariance matrix will also be implemented as well in the recursive MSWF. In summary, the required changes to the standard MSWF equations will include the following additions:

$$\mathbf{v}(l) = \mathbf{r}(l) - \mathbf{P}(l)\hat{\gamma}(l-1/l-1)$$

$$\hat{\gamma}(l/l) = \hat{\gamma}(l-1/l-1) + \mathbf{W}_1^{\mathrm{H}}\boldsymbol{\varepsilon}_1 \qquad (5.2)$$

$$\mathbf{K}_{\gamma}(l/l) = \mathbf{K}_{\gamma}(l/l-1) - \mathbf{W}_1^{\mathrm{H}}\boldsymbol{\xi}_\mathbf{1}\mathbf{W}_\mathbf{1}$$

Also, the initial conditions are again $\hat{\gamma}_0 = 0$ and $\mathbf{K}_{\gamma 0} = \sigma_{\gamma}^2\mathbf{I}$ as per the standard non-recursive MSWF implementation. In order to provide a better visualization to the structure of the recursive MSWF, they are shown on the following pages. Note that in both figures, $\hat{\gamma}_{11}$ is the same as $\hat{\gamma}(1/1)$ and $\mathbf{K}_{\gamma 11}$ is the same as $\mathbf{K}_{\gamma}(l/l)$. Also, instead of the measurement vector **r**(l) being used as $\mathbf{x_0}$, it is the innovation vector **v**(l) that will be used as $\mathbf{x_0}$ instead for each step of the recursive MSWF.

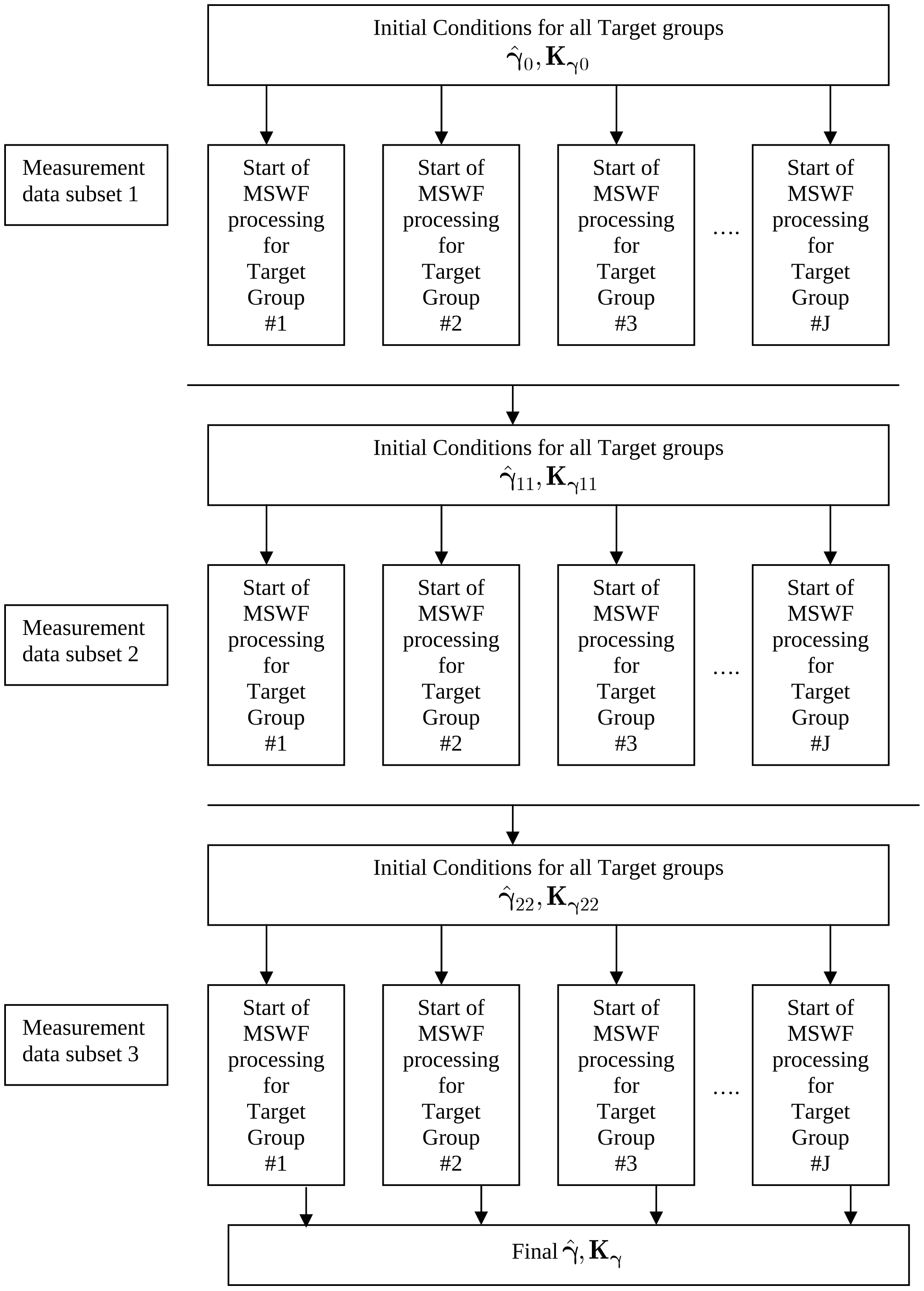
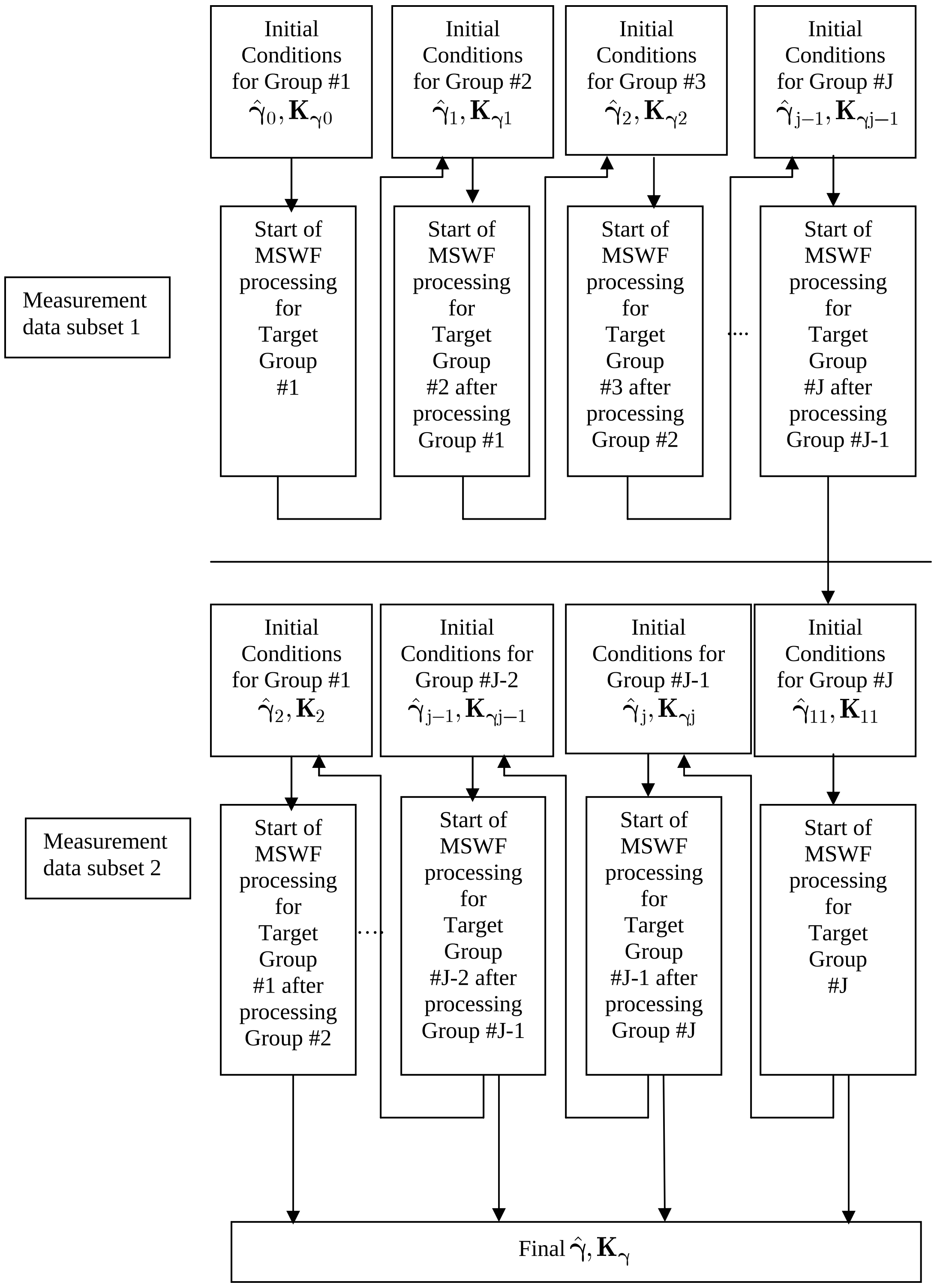

Initial Conditions for all Target groups
$\hat{\gamma}_0, \mathbf{K}_{\gamma 0}$
Measurement data subset 1
Start of MSWF processing for Target Group #1
Start of MSWF processing for Target Group #2
Start of MSWF processing for Target Group #3
….
Start of MSWF processing for Target Group #J
Initial Conditions for all Target groups
$\hat{\gamma}_{11}, \mathbf{K}_{\gamma 11}$
Measurement data subset 2
Start of MSWF processing for Target Group #1
Start of MSWF processing for Target Group #2
Start of MSWF processing for Target Group #3
….
Start of MSWF processing for Target Group #J
Initial Conditions for all Target groups
$\hat{\gamma}_{22}, \mathbf{K}_{\gamma 22}$
Measurement data subset 3
Start of MSWF processing for Target Group #1
Start of MSWF processing for Target Group #2
Start of MSWF processing for Target Group #3
….
Start of MSWF processing for Target Group #J
Final $\hat{\gamma}, \mathbf{K}_{\gamma}$

Figure 5-7: Recursive parallel MSWF Implementation



Initial Conditions for Group #1 $\hat{\gamma}_0, \mathbf{K}_{\gamma 0}$
Initial Conditions for Group #2 $\hat{\gamma}_1, \mathbf{K}_{\gamma 1}$
Initial Conditions for Group #3 $\hat{\gamma}_2, \mathbf{K}_{\gamma 2}$
Initial Conditions for Group #J $\hat{\gamma}_{j-1}, \mathbf{K}_{\gamma j-1}$
Measurement data subset 1
Start of MSWF processing for Target Group #1
Start of MSWF processing for Target Group #2 after processing Group #1
Start of MSWF processing for Target Group #3 after processing Group #2
….
Start of MSWF processing for Target Group #J after processing Group #J-1
Initial Conditions for Group #1 $\hat{\gamma}_2, \mathbf{K}_2$
Initial Conditions for Group #J-2 $\hat{\gamma}_{j-1}, \mathbf{K}_{\gamma j-1}$
Initial Conditions for Group #J-1 $\hat{\gamma}_j, \mathbf{K}_{\gamma j}$
Initial Conditions for Group #J $\hat{\gamma}_{11}, \mathbf{K}_{11}$
Measurement data subset 2
Start of MSWF processing for Target Group #1 after processing Group #2
….
Start of MSWF processing for Target Group #J-2 after processing Group #J-1
Start of MSWF processing for Target Group #J-1 after processing Group #J
Start of MSWF processing for Target Group #J
Final $\hat{\gamma}, \mathbf{K}_{\gamma}$

Figure 5-8: Recursive serial MSWF Implementation

Having completed the design implementations of both the recursive parallel and serial MSWF, the next task is to decide on the measurement data subset size to be used out of a total of 2856 measurement data as well as the target group size out of a total of 961 targets. From section 4.4.2, we know that the length of the measurement data subset cannot be smaller than the size of the target group used. As such, the decision is to use the combination shown in Table 5.1 for performing the experiment to determine whether is recursive MSWF faster than the standard MSWF and is there any tradeoff in this implementation.

Table 5.1: Measurement and Target Group Sizes for Recursive MSWF

Measurements per subset	Number of subsets	Target Group size = 961	Target Group size = 480	Target Group size = 320	Target Group size = 160
2856	1	√	√	√	√
1428	2	√	√	√	√
714	4	X	√	√	√
476	6	X	X	√	√

Now, in Table 5.1, some of the combinations are marked with the symbol “X”. The reason is because for these combinations, the size of the target group is bigger than the size of each measurement data subset. Thus it will not be possible to implement vector MSWF in these situations. Also, in row 1 of Table 5.1 where the number of measurements per subset is equal to 2856, it is equal to the parallel MSWF implementation since all measurement data are used at the same time and non recursion or iteration step is performed. Therefore, the results obtained from the combination in row 1 will be used as the benchmark to determine whether there is any benefit in using recursive MSWF over non recursive MSWF. After having decided on the combination of both measurement data subset and target group sizes to be used, a simulation is performed using these combinations and the results are discussed in the next section.

5.3.1 PERFORMANCE OF THE RECURSIVE MSWF

In the previous section, we have determined the combinations of the measurement data subset and target group sizes to be used in the simulation. As such, a simulation is executed for both the recursive parallel MSWF and recursive serial MSWF implementations. The results obtained are shown in Table 5.2 and Table 5.3 for the recursive parallel MSWF and Table 5.4 and Table 5.5 for the recursive modified serial MSWF implementation.

Table 5.2: Computational Timing Results for Recursive Parallel MSWF

Measurements per subset	Number of subsets	Target Group size = 961	Target Group size = 480	Target Group size = 320	Target Group size = 160
2856	1	113.203	241.112	222.498	215.547
1428	2	103.735	151.110	130.548	103.331
714	4	X	78.728	69.301	54.669
476	6	X	X	43.549	37.560

Table 5.3: Average Computed MSE for Recursive Parallel MSWF

Measurements per subset	**Number of subsets**	**Target Group size = 961**	**Target Group size = 480**	**Target Group size = 320**	**Target Group size = 160**
2856	1	-41.037	-41.037	-41.037	-41.037
1428	2	-41.037	-39.880	-39.350	-38.959
714	4	X	-37.093	-34.392	-30.052
476	6	X	X	-25.015	-21.524

From Table 5.2, we can observe that when the measurement data set is divided into more and more subsets, the computational time required will decrease significantly. Moreover, by looking at the corresponding entries in Table 5.3, we note to our pleasant surprise that the average computed *MSE* does not degrade at all in the case when all targets are included in one target group in conjunction to achieving this improvement in computational efficiency. However, in all other combinations, the improvement in the computational efficiency comes with a price which is the decrease in accuracy of the final estimated results. But in the case when the measurement subset is increased from 1 to 2 as seen from row 1 and row 2 of Table 5.2 and Table 5.3, this price to pay is very small when comparing with the amount of time savings that are achieved in the process. Thus, this is a situation where recursive parallel MSWF is able to meet our goal of improving the computational efficiency of the MSWF at very little cost.

Next, after further analyzing the results, we notice that as the target group size becomes smaller, the degradation in the accuracy becomes more severe. To explain the trend for the decrease in accuracy when the measurement data is divided into more subsets along with using target group size that is less than the total number of targets, we will need to examine the Error covariance matrix $\mathbf{K}_\gamma(l/l)$ that is generated after each iteration step. Essentially, $\mathbf{K}_\gamma(l/l)$ has a matrix size of 961×961. However, when the chosen target group size is less than 961, this will mean that at every iteration step, only the diagonal and near diagonal elements of $\mathbf{K}_\gamma(l/l)$ is updated since no information is available on the other elements that are away from the diagonal locations. To allow for better visualization, a sample of the $\mathbf{K}_\gamma(l/l)$ matrix using the recursive MSWF is shown below:

$$\mathbf{K}_\gamma(l/l) = \begin{pmatrix} x & x & x & o & o & o \\ x & x & x & o & o & o \\ x & x & x & o & o & o \\ o & o & o & x & x & x \\ o & o & o & x & x & x \\ o & o & o & x & x & x \end{pmatrix} \tag{5.3}$$

In equation 5.3 above, lets assume that there are a total of 6 targets and thus the $\mathbf{K}_\gamma(l/l)$ has a size of 6×6. Also, further assume that the targets are divided into 2 target groups with 3 targets per group. Using this scenario for implementing the recursive MSWF, only those locations marked with the symbol x will be updated after each iteration step whereas all the other locations marked with the symbol o will never be updated and their values will always remain as zero since the initial condition is $\mathbf{K}_{\gamma 0} = \sigma_{\gamma}^{2}\mathbf{I}$. Thus, $\mathbf{K}_{\gamma}(l/l)$ becomes a block diagonal matrix rather than a fully populated matrix and this will affect the accuracy of the estimation. Furthermore, if the size of the target group decreases, for example to 2 targets per group, it will make $\mathbf{K}_{\gamma}(l/l)$ to be more sparsely populated and the estimation errors will become more severe in this situation as seen from row 2 onwards in Table 5.3.

To further compound this problem, when the measurement data is divided into more subsets, one can imagine that the increase in the number of iteration steps will further aggravate the problem (number of times of error propagation also increases) and thus we can also see this trend from column 2 onwards in Table 5.3. Thus, after finding out the behavior of the recursive parallel MSWF, one will need to select the appropriate measurement data and target group sizes so as to achieve the desirable tradeoff between computational time and accuracy. As an example, choosing the combination of 4 measurement subsets along with a target group size of 480 provides a reasonable computed *MSE* of -37.093 dB with a computational time of 78.728 seconds. This constitutes to a time reduction of 162.384 seconds or 67% in time savings as compared to the original non recursive parallel MSWF implementation.

Now, after we have examined the results of the recursive parallel MSWF, we can conclude that the recursive parallel MSWF is indeed a good candidate to be considered for implementation when compared to the standard parallel MSWF implementation. In fact, in one situation, we can even achieve improvement in the computational efficiency at no cost at all in terms of the accuracy of results. The next task will be to look at the results obtained from the recursive modified serial MSWF that are shown in Table 5.4 and Table 5.5. Note that the modified serial MSWF is used instead of the standard serial MSWF because it has been shown in section 5.1 that the modified serial MSWF is a more efficient implementation.

Table 5.4: Computational Timing Results for Recursive Serial MSWF

Measurements per subset	**Number of subsets**	**Target Group size = 961**	**Target Group size = 480**	**Target Group size = 320**	**Target Group size = 160**
2856	1	113.203	305.190	445.894	879.208
1428	2	103.657	194.108	264.904	397.079
714	4	X	115.741	142.252	250.211
476	6	X	X	110.991	183.079

Table 5.5: Average Computed MSE for Recursive Serial MSWF

Measurements per subset	**Number of subsets**	**Target Group size = 961**	**Target Group size = 480**	**Target Group size = 320**	**Target Group size = 160**
2856	1	-41.037	-41.037	-41.036	-41.036
1428	2	-41.037	-39.812	-39.104	-38.565
714	4	X	-37.042	-32.768	-26.407
476	6	X	X	-22.622	-19.836

Looking at Table 5.4 and Table 5.5, we can observe the same trend in the recursive modified serial MSWF as in the recursive parallel MSWF. Thus, we can also conclude that the recursive modified serial MSWF is a good candidate for implementation as it is also able to achieve a significant reduction in computational timing in some situations along with non significant loss in accuracy of results. However, in general, the average computed *MSE* obtained from the recursive modified serial MSWF is usually worst off than that of the recursive parallel MSWF with the same combination after comparing the results between Table 5.3 and Table 5.5. Thus, in general, it is preferred to use recursive parallel MSWF if computing resources are not a factor of constraint.

At this moment, after trying out the few schemes as described in section 5.1 to 5.3, I believe that I have answered the question raised at the beginning of this Chapter on whether can the MSWF performance be improved in terms of the computational time required when we varies its implementation structure. After looking at new innovative approaches to both the original parallel and serial MSWF implementation, the answer to the question is a resounding “Yes”. Thus, I will also conclude the research work on the MSWF at this point since I believe that a very comprehensive investigation into the behavior of the MSWF has been performed from the start of Chapter 4 to this Chapter itself. I will then conclude on all my Thesis research work that has been performed in the concluding Chapter as well as giving some recommendations on future work that can be performed.

CHAPTER 6: CONCLUSIONS AND RECOMMENDATIONS

6.1 SUMMARY

In this Thesis, I have firstly presented the rationale behind using the Wiener filter for processing data obtained from a non-uniformly distributed aperture radar system as compared to using the Matched filter. However, one undesirable feature of the Wiener filter is that it requires performing a computationally expensive matrix inverse operation before the results can be obtained. Although Kalman filter is one technique that is able to avoid this matrix inverse operation in the Wiener filter, nevertheless it also has some pitfalls on this own. As such, I have looked at alternative techniques that can resolve the pitfalls of the Kalman filter along with additional techniques that are also more efficient than the Wiener filter.

In Chapter 3, I have investigated the use of the Square Root Covariance Filter (SRCF) as one alternative to the Kalman filter along with its reduced rank version, the Reduced Rank Square Root filter (RRSQRT). Both these techniques are found to achieve the same results as the Kalman filter when full rank processing is carried out. Moreover, they provide stability in the Error covariance matrix computation $\mathbf{K}_\gamma$ such that it will never end up having some negative Eigen values in it. On top of that, it has even been shown that the RRSQRT is able to achieve nearly the same level of accuracy of the final results as the Kalman filter when the rank has been reduced by up to 73%. For a small tradeoff in the results' accuracy, the rank reduction can even goes up to 95.8% as seen in the case when there are only 40 remaining Eigen vectors in $\mathbf{K}_\gamma$ as compared to an initial value of 961 Eigen vectors. Thus, this can constitute to significant savings in computing resources during implementation. Although the need for Eigen decomposition operation to achieve the rank reduction increases the computational timing of the RRSQRT such that it exceeds that of the Kalman filter, however the increase is not tremendous and it is believed that this shortcoming can be overcome when there is a more efficient method to perform the Eigen decomposition operation.

Next, besides the square root filters, I have also looked at another technique that is more efficient that Wiener filter in its implementation while at the same time, easily allows rank reduction measures to be performed. This technique is known as the Multi-Stage Wiener filter (MSWF) based on orthogonal projections. After deriving the equations and implementation of the MSWF required for our radar problem domain, I am able to show that this filter is able to achieve the same results with the Wiener filter even when less than full rank processing is performed. Although the MSWF is not as computationally efficient as the Kalman filter, its performance gap with the Kalman filter does not degenerate even when the problem statement expands in size. For instance, in the case of using 3060 measurements for 961 targets, the fastest timing achieved by the MSWF is 112.572 seconds as compared to 35.262 seconds for the Kalman filter (refer to Table 4.6 and 4.10). This is equivalent to a ratio of about 3.2. As for the case of using 11460 measurements for 3969 targets, the fastest timing achieved by the MSWF is 3432.5 seconds as compared to 1192.1 seconds for the Kalman filter (refer to Table 4.11). This is equivalent to a ratio of about 2.9. Thus, the performance gap remains essentially constant. However, the performance gap between the MSWF and the Wiener filter widens from a value of 1.65 (186.078 seconds for Wiener filter versus 112.572 seconds for MSWF) using the smaller data set to a value of 2.72 (9349.3 seconds for Wiener filter versus 3432.5 seconds for MSWF) when using the larger data set. Thus, the advantage of the MSWF over the Wiener filter becomes more prominent as the data set increases.

Besides achieving some meaningful results using the MSWF, I have also shown that it is possible to further improve its computational efficiency when care is taken to group the targets according to the criteria of high cross correlation magnitudes or when the MSWF is implemented in a recursive manner. Although no numerical figures are available in the case of using the target grouping approach due to the imperfection of my grouping schemes, nevertheless the overall trend points to the high probability of performance enhancement. Thus, these findings will be useful to future researchers who will need to implement the MSWF for their needs.

6.2 RECOMMENDATIONS FOR FUTURE WORK

In all my findings and results obtained for the different filters that are introduced in the Thesis research, the common shortcoming that needs further improvement is in the speed of the computation of these filters when compared to the Kalman filter. When we examine the case of the Reduced Rank Square Root filters, it is the Eigen decomposition operation that is the main culprit for the computational inefficiency. Therefore, one area that we can further look into is in finding faster ways of performing Eigen decomposition so that the total timing taken by the RRSQRT will not exceed that of the Kalman filter.

Besides that, we may also want to look into other SAR scenarios where it is only required to know the scattering coefficients of a few disjoint smaller areas of interest but not the whole illumination area. In these situations, we can then start the iteration right away with a reduced rank version of the square root covariance matrix $\mathbf{S}(0/0)$ for the RRSQRT whereas the Kalman filter will still need to work on the full size of the Error covariance matrix $\mathbf{K}_{\gamma}$. As a result of starting with a smaller matrix size of $\mathbf{S}(0/0)$, it may end up reducing the time taken by the subsequent Eigen decompositions to a point that the RRSQRT may outperform the Kalman filter at the end of the iterations. However, in such scenarios, the tradeoff will be that no estimation or information will be obtained from those regions that are designated as "don't care" in the whole illumination area.

Next, coming to the implementation of the Multi-Stage Wiener filter, one area that future researchers can start looking into is the optimal grouping scheme that will group targets together either using the criteria of high cross correlation or vice versa. Once the optimal grouping scheme is developed, it can then be tested on either the non recursive or recursive MSWF implementations to verify whether the computation speed goes up further in both situations, i.e. using less stages of decomposition. Also, besides looking into this area , another possibility will be to re-look at the present structure of the MSWF that requires a forward iteration step, the turn around step and the backward iteration step. In the last couple of years, there is an emergence of a new type of MSWF that is based on Conjugate Gradients rather than on Orthogonal Projections as seen in [28] and [29] etc. For the Conjugate Gradient based MSWF (MSWF-CG), there is only a need to perform the forward iteration step in its structure. As such, it may be a more efficient form of MSWF as compared to the current MSWF that is based on orthogonal projections. Thus, future work may involve the adaptation of the MSWF-CG algorithm to our radar scenario before running simulations to measure its performance as compared to the current MSWF and the Kalman filter.

6.3 CONCLUSIONS

The focus of this thesis is on developing reduced rank versions of the conventional Wiener or MMSE filter such that these reduced rank implementations will be much more efficient than the Wiener filter while only sacrificing small loss in the final accuracy of the results. From the findings presented in the thesis, this objective has been fully met. Although the techniques developed in this thesis is not as efficient as the Kalman filter which is another efficient implementation of the Wiener filter, nevertheless the performance gap between the former and the latter is not very significant. Moreover, the RRSQRT will be able to overcome some potential pitfalls that are documented in the Kalman filter. With additional research efforts invested into the recommendations that are identified in the previous sections, it may result in the complete elimination of the performance gap between the reduced rank filters and Kalman filters.

Finally, the research work involved in this thesis have given me a great appreciation of the domain of linear filtering as well as the fundamentals of radar system design modeling. It is believed that this knowledge will aid me greatly in my subsequent professional career down the years.

REFERENCES

[1] Nathan Goodman, “SAR and MTI processing of sparse satellite clusters”, Doctoral thesis, The University of Kansas, July 2002.

[2] Peter S. Maybeck, “Stochastic Models, Estimation and Control”, Volume I, Academic Press 1979, Chapter 7.

[3] R.W. Stewart and R. Chapman, “Fast Stable Kalman Filter Algorithms utilizing the Square Root”, International Conference on Acoustics, Speech and Signal Processing, April 1990, volume 3, pg 1815-1818.

[4] J. Scott. Goldstein and Irving S. Reed, “A New Method of Wiener Filtering and its Application to Interference Mitigation for Communications”, IEEE MILCOM Nov 97 Proceedings, volume 3, pg 1087-1091.

[5] J. Scott. Goldstein and Irving S. Reed, “A Multistage Representation of the Wiener Filter Based on Orthogonal Projections”, IEEE Transactions on Information Theory, volume 44, No 7, Nov 1998.

[6] Subhash Gullapalli, ”Application of Kalman Filtering Technique for SAR Processing of sparse satellite clusters”, Master’s Thesis, The University of Kansas, December 2002.

[7] J.M. Stiles, “Determination of Numeric Model Parameters”, Revision A, The University of Kansas, Radar Systems and Remote Sensing Laboratory, Lawrence, KS, July 2004

[8] J.M. Stiles, “Space-Time Radar Transmission, Target and Measurement Model”, Revision E, The University of Kansas, Radar Systems and Remote Sensing Laboratory, Lawrence, KS, August 2004

[9] Jim Stiles, Vishal Sinha, Atulya Deekonda, “Optimal Space-Time Transmit Signals for Multi-Mode Radar”, The University of Kansas, Radar Systems and Remote Sensing Laboratory, Lawrence, KS, Nov 2005

[10] Simon Haykin, “Adaptive Filter Theory”, Fourth edition, Prentice Hall 2002, Chapter 2.

[11] Mohinder S.Grewal, Angus P. Andrews, “Kalman Filtering”, 2nd Edition, Wiley Interscience, 2001, Chapter 6.

[12] M. Verlaan, A.W. Heemink, “Reduced Rank Square Root Filters for Large Scale Data Assimilation Problems”, Second International Symposium on Assimilation of Observations in Meteorology and Oceanography, World Meteorological Organization, Mar 1995, pg. 247-252.

[13] M. Verlaan, A.W. Heemink, “Tidal Flow Forecasting using Reduced Rank Square Root Filters”, Stochastic Hydrology and Hydraulics, volume 11, no 5 1997, pg 349-368.

[14] M. Verlaan, A.W. Heemink, “Convergence of the RRSQRT Algorithm for Large Scale Kalman Filtering Problems”, Delft University of Technology, 1997, Technical report 97-19.

[15] G. Golub, C. Van Loan, “Matrix Computations”, John Hopkins University Press, 2nd edition, 1989, pg 427.

[16] J. Scott. Goldstein and Irving S. Reed, “Reduced Rank Adaptive Filtering” IEEE Transactions on Signal Processing, 1997, volume 45, pg 492-496.

[17] Michael L. Honig, J. Scott. Goldstein, “Adaptive Reduced-Rank Residual Correlation for DS-CDMA Interference Suppression”, IEEE Conference on Signals, Systems and Computers, 1998, volume 2, pg 1106-1110.

[18] Michael L. Honig, J. Scott. Goldstein, “Adaptive Reduced-Rank Interference Suppression Based on the Multi-Stage Wiener Filter”, IEEE Transactions on Communications, 2002, volume 50, pg 986-994.

[19] Michael L. Honig, Weimin Xiao, “Performance of Reduced-Rank Linear Interference Suppression”, IEEE Transactions on Information Theory, 2001, volume 47, pg 1928-1946.

[20] J. Scott. Goldstein, Irving S. Reed, Dan E. Dudgeon and Joe R. Guerci, “A Multistage Matrix Wiener Filter for Subspace Detection”, IT workshop on Detection, Estimation, Classification and Imaging, Santa Fe, NM, USA, Feb 24-26, 1999.

[21] J.M. Stiles, “Implementation of the Multi-Stage Wiener Filter”, Revision B, The University of Kansas, Radar Systems and Remote Sensing Laboratory, Lawrence, KS, Jan 2006.

[22] J. Scott. Goldstein, Irving S. Reed, “Theory of Partially Adaptive Radar”, IEEE Transactions on Aerospace Electronic System, 1997, volume 33, pg 1309-1325.

[23] Ricks, D.C., J. Scott. Goldstein, “Efficient Architectures for Implementing Adaptive Algorithms”, Proceedings of the 2000 Antenna Applications Symposium, Allerton Park, Monticello, Illinois, Sep 20-22, 2000, pg 29-41.

[24] Seema Sud, Wilbur L. Myrick, J. Scott Goldstein, Michael D. Zoltowski, ”Performance Analysis of a Reduced Rank MMSE MUD for DS-CDMA”, GlobalComm 2001, volume 5, pg 3158-3162.

[25] Seema Sud, Wilbur L. Myrick, Paula Cifuentes, J. Scott Goldstein, Michael D. Zoltowski, “A Low Complexity MMSE Multi-user detector for DS-CDMA”, 35th Asilomar Conference on Signals, Systems and Computers, Nov 2001, volume 1, pg 404-409.

[26] Seema Sud, Wilbur L. Myrick, Paula Cifuentes, J. Scott Goldstein, Michael D. Zoltowski, “Reduced Rank Matrix Multi-stage Wiener Filter with applications in MMSE Joint Multi-user detection for DS-CDMA”, IEEE International Conference on Acoustics, Speech and Signal Processing, 2002, volume 3, pg 2605-2608.

[27] Harry L. Van Trees, “Optimum Array Processing, Part IV of Detection, Estimation and Modulation Theory”, Wiley Interscience, 2002, 1st edition, pg 505 to 510.

[28] Hongya Ge, M. Lundberg, Louis L. Scharf, “Reduced-Rank Multi-user Detectors Based on Vector and Matrix Conjugate Gradient Wiener Filters”, IEEE 5th workshop on Signal Processing Advances in Wireless Communications, 2004, pg 189-193

[29] Zhi Tian, Hongya Ge, Louis L. Scharf, “Low-Complexity Multi-user Detection and Reduced-Rank Wiener Filters for Ultra-Wideband Multiple Access”, International Conference on Acoustics, Speech and Signal Processing, 2005, pg 621-624.