FINAL REPORT

Aaron Sanders

Bachelor of Engineering(Honors) with a Degree in Science
Electronics and Physics

MACQUARIE
University
SYDNEY·AUSTRALIA

School of Engineering
Macquarie University

November, 2017

Supervisor: Ediz Cetin

ACKNOWLEDGMENTS

I would like to acknowledge the following people in no particular order for there help through the course of my undergraduate degree and research thesis. Firstly my supervisor Ediz Cetin for your guidance and support in this project. To John Tuthill at CSIRO for giving me this project to work on. To my loving partner Anwyn for all you help and tolerance during these last few years. To my parents for giving me every opportunity in life to help me achieve my goals. To Bob and Amanda Shoemark for there support and assistance during this project. I would also like to thank my fellow unit staff and students for making this time as fun as possible.

STATEMENT OF CANDIDATE

I, Aaron Sanders, declare that this report, submitted as part of the requirement for the award of Bachelor of Engineering in the School of Engineering, Macquarie University, is entirely my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualification or assessment an any academic institution.

Student's Name: Aaron Sanders

Student's Signature: Aaron Sanders(electronic)

Date:6/11/2017

ABSTRACT

With the adoption of Phased Array Feeds on telescopes such as the Australian Square Array Pathfinder comes problems in processing data coming from the array. One such problem is the calculation of beam weights for beam formers. A need has been identified for a fast matrix inversion algorithm capable of supporting these data rates. A hardware implementable algorithm addressing the matrix inversion problem will be presented along with it limitations and advantages. Several other algorithms were also tested for suitability with the results presented.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years there has been a move by the radio astronomy community to adopt Phased Array Feeds for receiving radio signals. A number of engineering test beds have been commissioned, with some currently being used for astronomical science [1]. Phased arrays were first demonstrated as far back as 1905 by Karl Ferdinand Braun. Phased array antenna technology was pioneered by RADAR users starting as early as World War II. Phased array feeds are now present in most RADAR and SONAR systems used by both civilian and military RADAR today. The motivation for the adoption of PAF technology comes from the ability for a PAF to have any arbitrary beam shape, and for that beam to be electronically steered across the Field of View. This design has many advantages over traditional fixed beam antenna.

Traditional fixed beam antenna consisted of a dipole(or similar) antenna, with a shaped reflector, and wave guide that provided high gain, and a good impedance matching between free space and a Low Noise Amplifier. The LNA which in some cases is cryogenicly cooled to minimize system noise [2], provides the first stage amplification. The quality of the LNA and the beam optics are critical factors in determining the Signal to Noise Ratio of the receiver. Due to the low power of the Signals Of Interest a high gain reflector and antenna is needed. A single dipole antenna is non directional resulting in a large amount of unwanted signal being present from other Radio Frequency Interfering sources, in the case of radio astronomy this includes aircraft, satellites, CB radio and mobile phone towers as well an electrical noise from sources such as microwaves [3].

While the fixed beam antenna used in radio astronomy provide high gain, they suffered from having a limited Field Of View resulting in a large number of pointing's to be taken to form a picture of the sky as part of a process called Mosaic [4]. The Parkes multi beam receiver [5] was developed to help address this issue. It consisted of 13 dipole antenna with a feed horn forming the beam shape allowing for 13 pointing's to be conducted at the same time. However the multi beam receiver still required a shaped reflector to move the beams into position.

Telescopes such as the Australian Telescope Compact Array use a different technique call interferometry [6]. Informatory is another form of radio astronomy that takes an array of single fixed beam antenna, and electronically joins their signals together to form an image at much higher resolution. This gave two advantages, firstly a high signal gain as stated previously. Secondly it provided a higher resolution image through line delays form each receiving element and correlation of signals. Fixed beam interferometers survey speed is still limited by the slew rate of the antenna. The slew rate is determined by the size of the telescope. Large telescopes such as the ones used at the ATCA take some time to move the 350 tonne antennas into position. The accuracy needed in pointing also effects the speed of slew, A telescope requiring high accuracy pointing's is slower then a communication antenna which just needs to point roughly where the signal is. This limitation made interferometry using fixed beams unsuited to applications such as radar and sonar however were well suited to radio astronomy. Complexity associated with telescopes requiring good pointing accuracy adds significant cost and design challenges when implementing fixed beam antenna [7]

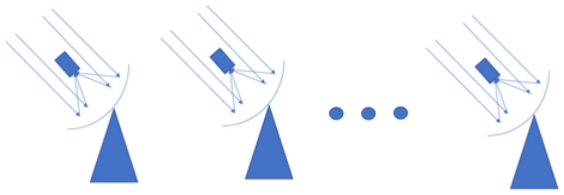


Figure 1.1: Fixed beam interferometer receiving signal form SOI. The outputs of each receiver will pass through a correlation algorithm

To obtain a high degree of pointing accuracy as well as having a fast slew rate required the use of Phased Array Feeds to be deployed for RADAR and SONAR systems. A PAF consists of several non-directional antenna placed in a 1D or 2D array. The output of each antenna is fed into beam former which applies weights to each antenna's output then sums the outputs. This electronically generated beam can be changed by simply changing the values for the weights. This means that beams can be moved electronically across the sky at speeds limited only by the speed at which the weights can be calculated.

The radio astronomy community has began to adopt the PAF technology on telescopes such at the Australian Square Kilometer Array Pathfinder project [8]. The objective for the PAF technology being implemented in radio astronomy is to generate up to 9 electronic beams on each antenna to allow for an optimized FOV though the use of mosaic processes. The optimized FOV will have several features such as higher sensitivity on sources of interest, mosaic beams for large sky coverage provide an ability for fast sky surveys to be conducted. RFI mitigation can be achieved by moving nulls in the beams FOV to the position of strong sources of RFI resulting in the RFI signals having a low gain applied to them when compared to the gain applied to the SOI.

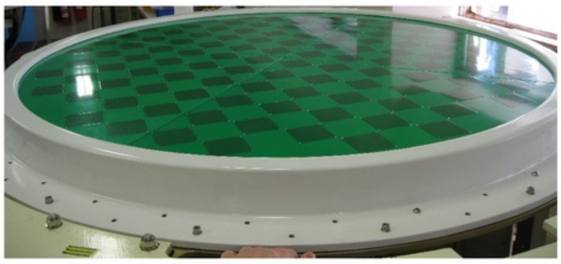


Figure 1.2: Early prototype PAF antenna patches for the CSIRO ASKAP project[image:Aaron Sanders]

1.1 Motivation

As FPGA technology improves, the ability to perform more complex mathematical processes on signals open up opportunities for radio astronomers to implement new signal processing techniques. Several techniques have been proposed for active RFI mitigation, or removal using new signal processing techniques [3]. Active RFI mitigation and removal is becoming more important as the microwave spectrum become more congested due to the increased demand for wireless communications at higher and higher bandwidths and frequencies.

These new techniques, designed to combat RFI, require fast mathematical operators capable of being integrated into existing, or new telescope architecture. The motivation for this thesis starts here. For these new techniques to be implemented in the ASKAP and other similar telescopes, a range of mathematical operators need to be developed for implementation in FPGA based environments, and be capable of operating on the large data sets generated by these telescopes.

The goal of this project is to start the process of developing a fast beam former weight calculation algorithm for implementation in a FPGA architecture such as the one used in ASKAP. This is quite a large project and would be to large for an undergraduate engineering research thesis scope in it's entirety.

1.2 Project Goals

Several goals for this project have been identified based on communication with the client, CSIRO Astronomy and Space Science group. These goals have been selected as they offer a problem of suitable size for a thesis. The goals have also been selected based on the need and potential impact to current and future system development.

The goal of the project is to research and develop an algorithm for matrix inversion suitable for deployment in an FPGA based environment. A break down of the goals include

- Identify potential algorithms to perform fast matrix inversion
- Verify the identify algorithms are numerically acceptable
- Develop and test the algorithms in MATLAB
- Develop and test the algorithms suitability for FPGA based deployment

The remainder of this report will be structured as follows. The remaining of this chapter will be the project plan, scope and budget. Chapter two a review of the literature associated with the proposed project will be presented. Chapter three will present information regarding the system design and algorithm verification process. The system testing and verification is presented in chapter four. A review of the results will be presented in chapter 5. The recommendation of this report will be presented in chapter 6. The conclusion will be presented in chapter 7.

1.3 Project Plan

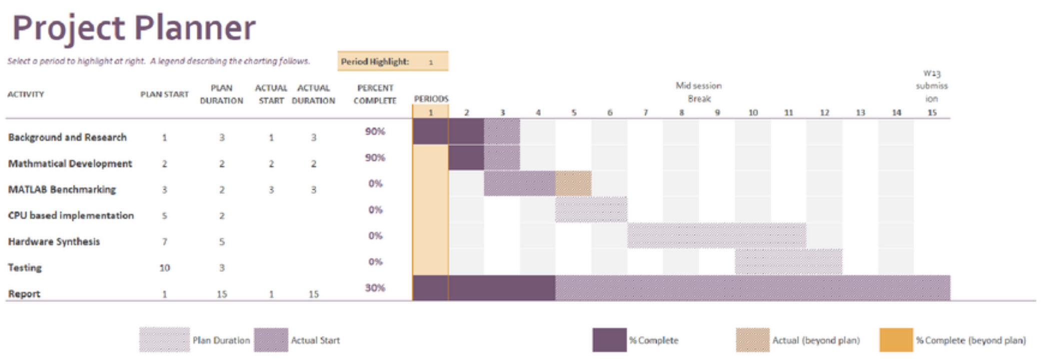
1.3.1 Scope

Due the the size of the problem it is not practical to develop the full beam former weight calculator during the 13 week engineering thesis program. It was decided through consultation with the CSIRO astronomy and spaces sciences division, CASS, to focus the matrix inversion problem for the ENGG411 project. This particular problem was chosen as it has the ability to be integrated into existing systems and offer improvements without having to develop the rest of the system. It is also a fundamental mathematical function that is require for use in other more advanced algebraic and statistical operators used in signal processing.

The project will involve the development of a mathematical algorithm that can be tested for initial performance verification. If time permits a test version of the algorithm will be developed for deployment on the Zynq MicroZED development board from AVNET.

1.3.2 Time Management

The project evolved from the initial project plan during the course of the Thesis. As more potential algorithms were found and problems were encountered with some algorithms it was decided to re-focus the project. After some consultation with CSIRO the main priority of the project was determined to be a survey of existing matrix inversion algorithms. From the research into existing methods a new or ideal method suitable for use with the ASKAP antenna would be identified and proposed for development. The gantt chart in 1.4 represent the original time line for the project.

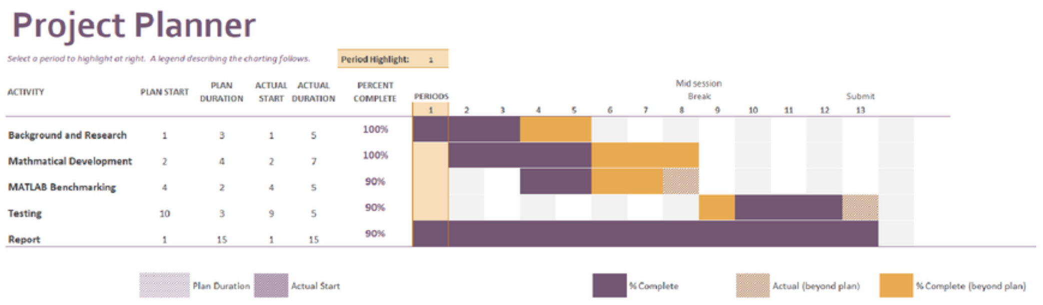

Project Planner

Select a period to highlight at right. A legend describing the charting follows.

Period Highlight: 1

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Background and Research	1	3	1	3	90%
Mathmatical Development	2	2	2	2	90%
MATLAB Benchmarking	3	2	3	3	0%
CPU based implementation	5	2			0%
Hardware Synthesis	7	5			0%
Testing	10	3			0%
Report	1	15	1	15	30%

Figure 1.3: Gantt summary of progress on project as of 4/9/17


Project Planner

Select a period to highlight at right. A legend describing the charting follows.

Period Highlight: 1

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Background and Research	1	3	1	5	100%
Mathmatical Development	2	4	2	7	100%
MATLAB Benchmarking	4	2	4	5	90%
Testing	10	3	9	5	90%
Report	1	15	1	15	90%

Figure 1.4: Gantt summary of progress on project as of 6/11/17

It was important to have flexibility during this project as many different avenues of research were explored with many leading to dead ends. As it can be seen the project plan changed quite a bit during the course of the project. This was planed for so did not cause any problems for the project.

1.3.3 Budget

As this project is primarily a simulation and coding problem the cost is expected to be minimal. The development board was provided by CASS. All the software tools are already licensed by the university and are freely available to students. All the remaining parts and tools needed were already purchased as part of past projects or university courses and so contribute no cost to this project. A full breakdown of cost is shown in table 1.1.

Table 1.1: Project cost breakdown

Item	Reason	Provider	Cost	Cost to project
Computer	Running simulations	Personal	$2500	$0
Matlab	Simulations and modeling	University	$3000	$0
Vivado	FPGA design	Personal	FREE	$0
Zynq	FPGA/ARM test bench	CSIRO	$199	$0
JTAG bugger	Programing/debugging FPGA	Personal	$160	$0

The cost of the project did not change during the course of the project. There were no added expenses that were incurred.

Chapter 2

Background and Literature Review

Beam forming and matrix inversion are two large and very complex field with research going back hundreds of years in come cases. With this in mind it would be impractical for an in depth coverage of all relevant theory to be presented here. The following chapter will provide the fundamental knowledge relating to this thesis. The overview will start with an detailed description of beam forming practices on the ASKAP array. Then a overview of matrix inversion will be presented as well as methods of estimating the inverse of a matrix. Finally a survey of existing approaches to performing matrix inversion in FPGA's will be presented.

2.1 Beam Forming

In adaptive beam forming systems such as the Phased Array Feed in the ASKAP project, an array of N antenna are spaced in either a 1D or 2D arrangement separated by some distance d. Each antenna has a non-directional beam shape allowing it to have a large FOV with a fairly stable gain across the FOV. The SOI will arrive as a wave front assuming the distance from the array to SOI is much greater then the distance between elements in the array.

The output of each of the N antenna are sampled at an appropriate frequency for the Signal Of Interest and produces a $N \times 1$ vector of amplitudes $x[n]$ where n is the time index. The signal will consist of two parts $\mathbf{V}_s$ which is the normalized array response to a far field SOI. The second part is the noise associated with the system and signal from sources out side the field of interest $\mathbf{n}$.

$$\mathbf{x}[n] = \mathbf{V}_s s[n] + \mathbf{n}[n] \tag{2.1}$$

To produce a single output signal all of the N outputs are multiplied by a weight and then added to give the output signal $y[n] = wx[n]$. The weight values w is a $N \times 1$ vector

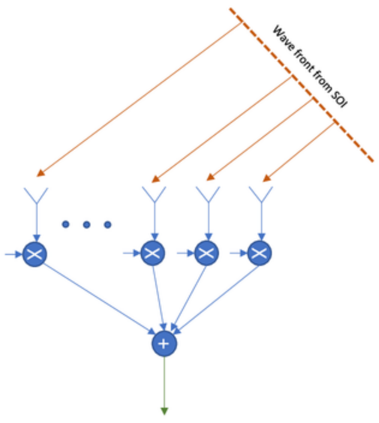


Figure 2.1: Wavefront delay from a point source on PAF receiver

of predetermined weights. The weights enable the system to add and cancel signals from set antenna allowing for constructive and destructive addition of signals and thus forming a electronically controlled beam.

Many methods of weight calculation exist including

- Least Mean Squared Algorithm
- Sample Matrix Inversion Algorithm
- Recursive Least Square Algorithm
- Conjugate Gradient Method
- Max Signal to Noise Ratio Method
- Constant Modulus Algorithm
- Numerically Optimized Method

All methods are computationally expensive and often are not practical for large arrays needing real time processing. Several approaches such as the maxSNR, Recursive Least Square Algorithm and Sample Matrix Inversion Algorithm require matrix inversion as part of the algorithm.

2.2 Beam Weight Calculation

The CSIRO ASKAP project performance requirements specified that the maxSNR method was being used to calculate the beam former weights [9]. As the end goal for this project is to design a algorithm suitable for hardware based implementation of algebraic operators for ASKAP the focus will be on algorithms suitable for implementation on ASKAP hardware. The algorithm must also offer comparable results to the maxSNR method. The target end user platform for the project is the "Redback DSP Board" shown in the figure below. The read back board consists of 6 Xilinx FPGA's currently able to process 48MHz bandwidth(8MHz per FPGA). There is a total of 288 of these units in use in the ASKAP array.

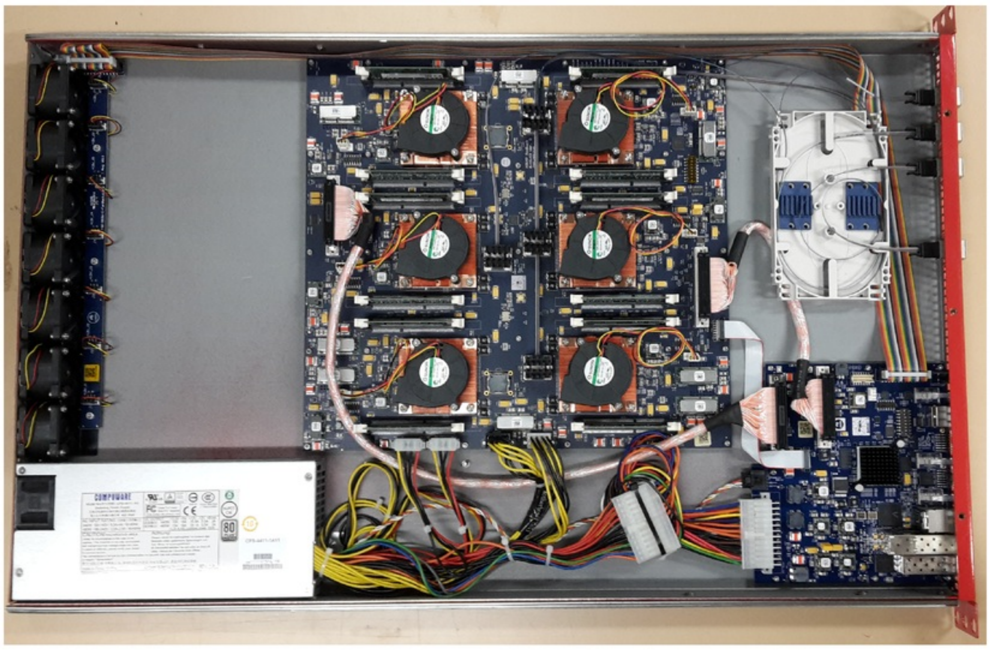


Figure 2.2: The Redback DSP board source:John Tuthill, CSIRO

2.2.1 maxSNR

The maximum Signal to Noise Ratio algorithm is currently the algorithm of choice in the PAF Radio Astronomy community as it easy to implement and offers a simple method for calibration [10]. It relies on a single on-source and a single off-source measurement to calculate the required beam weights for the PAF. This calibration procedure has the advantage of providing the optimal SNR for the system, eliminating system related noise(assuming minimal short term variances of noise). The maxSNR method is refereed to as a fixed-adaptive beam as the beam pattern will remain fixed for days to months at a time. The maxSNR method does not adapt easily to fast changing sources of noise such as aircrafts. The ASKAP array being designed by CASS currently uses the maxSNR method for beam former weight calculations [8]. For future PAF designs it is desired to develop a new type of beam former that is capable of combating RFI though techniques such as null point steering that are currently not possible with the maxSNR method.

2.3 Matrix Inversion

The inverse of a square matrix **A** with dimensions $n \times n$ is a matrix **B** of the same dimensions of **A**. The inverse matrix has the properties **AB=I** and **BA=I** where **I** is the identity matrix. A square matrix is not required to have a inverse, but in cases where it does then the inverse in unique [11].

2.3.1 LU-Decomposition

One of the most common methods of performing a matrix inverse in a digital computer is the LU-Decomposition where LU stands for Lower and Upper. For a square matrix **A** a LU-Decomposition is performed by first defining

$$\mathbf{LU} = \mathbf{A}$$

where **L** and **U** are the upper and lower triangle of the matrix **A**. The linear set

$$\mathbf{Ax} = (\mathbf{LU})\mathbf{x} = \mathbf{L}(\mathbf{Ux}) = b$$

can then be solved using forward and backward substitution which is known to be an efficient process.

$$\begin{aligned} y_1 &= \frac{b_1}{\alpha 11} \\ y_i &= \frac{1}{\alpha ii}[b_i - \sum_{j=1}^{i-1} \alpha_{ij} y_j] i = 2, 3, ..., N \qquad (2.2) \\ x_N &= \frac{y_n}{\beta nn} \\ x_i &= \frac{1}{\beta_{ii}}[y_i - \sum_{j=i+1}^{n} \beta_{ij} x_j] i = n-1, N-2, ...1 \qquad (2.3) \end{aligned}$$

This method offers similar performance for finding A^{-1} when compared to the Gauss-Jordan method. The reason for the LU-Decomposition methods dominance comes in solving linear equations such as $A^{-1}B$ which requires one less matrix multiplication and is more accurate. For the LU method the total number of executions in the summation loop is $\frac{1}{6}n^3$ for the forward substitution and $\frac{1}{2}n^3$ for the backwards substitutions [12].

2.3.2 Sherman-Morrison-Woodbury formula

The Sherman-Morrison-Woodbury formula is a special case of the Woodbury identity. The Woodbury identity states

$$(\mathbf{A}+\mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1} \tag{2.4}$$

Where $\mathbf{A}$ is a $n \times n$ matrix, $\mathbf{U}$ is a $n \times k$ matrix, $\mathbf{C}$ is a $k \times k$ matrix and $\mathbf{V}$ is a $k \times n$ matrix. This identity can be applied to the case where $\mathbf{A}^{-1}$ is already known and it is desired to update it with $(\mathbf{A}+\mathbf{UCV})^{-1}$. For the special case where $\mathbf{C}$ is a 1×1 unit matrix is know as the Sherman-Morrison formula. The Sherman-Morrison formula

$$(\mathbf{A}+\mathbf{UV}^{T})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{I}+\mathbf{V}^{T}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^{T}\mathbf{A}^{-1} \tag{2.5}$$

where $A \in \mathbb{R}^{n\times n}$ and $\mathbf{U},\ \mathbf{V}$ are $n \times k$. The update of the inverse by a rank k correction will result in a rank k correction to the inverse [13].

In 1950 Sherman and Morrison presented a special case of the Woodbury identity [14]. In their paper a method for a rank-1 update was presented. For $a_{ij},\ i = 1, 2, ..., n$, $j = 1, 2, ..., n$ giving a square matrix a of size n. $b_{ij} = a^{-1}$. A_{ij} denotes the elements of A which differ from a in only one element A_{RS}. B_{ij} is the elements of the inverse of A.

It was shown that the B could be computed from Δa_{RS} and b.

$$\begin{aligned} A_{RS} &= a_{rs} + \Delta a_{RS} \\ B_{rj} &= b_{rj} - \frac{b_{rR} b_{Sj} \Delta a_{RS}}{1 + b_{SR}\Delta a_{RS}}\ r = 1, 2, ..., n\ j = 1, 2, ..., n \end{aligned} \tag{2.6}$$

which can be split into three cases $r = S$, $j = R$, and all others

$$B_{Sj} = \frac{b_{Sj}}{1 + b_{SR}\Delta a_{RS}} j = 1, 2, ..., n \tag{2.7}$$

$$B_{rR} = \frac{b_{rR}}{1 + b_{SR}\Delta a_{RS}} j = 1, 2, ..., n \tag{2.8}$$

using 2.8 we get 2.6 in the following form

$$B_{rj} = b_{rj} - B_{rR} b_{Sj} \Delta a_{RS} \; r = 1, 2, ..., S-1, S+1, ..., n \; j = 1, 2, ..., R-1, R+1, ..., n \tag{2.9}$$

which shows that the elements of B contained in the Sth row and Rth column are directly proportional to the corresponding elements of b. It was also commented that should a matrix change by more than one element that repeated applications of this technique could be applied to get to the correct result. This technique was then adapted and applied to a matrix with rank-n updates required [15]. The updated algorithm was designed for use of a Cray super computer and so was designed for parallel computations.

2.4 Existing methods and implementations

Given an input autocorrelation matrix of the form

$$\mathbf{R}_{xx} = \sigma^2 \mathbf{I} + \sum_{i=1}^{N} u_i u_i^H \tag{2.10}$$

Where σ^2 is the noise power and u is the input vector from an array of N elements and c^H is the transpose-conjugate of the input vector form. The Sherman-Morrison formula can then be expressed in terms that match our application

$$(\mathbf{A}^{-1} + \mathbf{u}\mathbf{v}^H)^{-1} = \mathbf{A}^{-1} - \frac{(\mathbf{A}^{-1}\mathbf{u}_i \mathbf{v}_i^H \mathbf{A}^{-1})}{1 + \mathbf{v}^H \mathbf{A}^{-1} \mathbf{u}} \tag{2.11}$$

and by using the same notation as (2.6) we get

$$\mathbf{R}^{-1} = (\mathbf{R}_{i-1}^{-1} + \mathbf{u}_i \mathbf{u}_i^H)^{-1} = \mathbf{R}_{i-1}^{-1} - \frac{(\mathbf{R}_{i-1}^{-1} \mathbf{u}_i \mathbf{u}_i^H \mathbf{R}_{i-1}^{-1})}{1 + \mathbf{u}_i^H \mathbf{R}_{i-1}^{-1} \mathbf{u}} \tag{2.12}$$

It can be seen that this form of the formula will present a computational problem due to the division. This has been overcome by the introduction of a scaling factor α such that

$$\begin{aligned}\alpha_i \mathbf{R}^{-1} &= (\alpha_{i-1} + \mathbf{u}_i^H \alpha_i \mathbf{R}_{i-1}^{-1} \mathbf{u}_i)(\alpha_i \mathbf{R}_{i-1}^{-1} + \frac{\mathbf{u}_i \mathbf{u}_i^H}{\alpha_{i-1}})^{-1} \\ &= \mathbf{R}_{i-1}^{-1}(\alpha_{i-1} + \mathbf{u}_i^H \alpha_{i-1} \mathbf{R}_{i-1}^{-1} \mathbf{u}_i) - (\alpha_{i-1} \mathbf{R}_{i-1}^{-1} \mathbf{u}_i \mathbf{u}_i^H \alpha_{i-1} \mathbf{R}_{i-1}^{-1}) \qquad (2.13)\end{aligned}$$

where α takes the following definitions. $\alpha_0 = 1$ and

$$\alpha_{i+1} = \alpha_i(\alpha_i + \mathbf{u}_{i+1}^H \alpha_i \mathbf{R}_i^{-1} \mathbf{u}_{i+1}) \qquad (2.14)$$

It has been shown that using this result, that a matrix inversion algorithm could be implemented without the use of operators such as division and square roots, dramatically reducing the circuitry complexity of an FPGA implementation [16]. This is a critical result allowing for the development of an efficient FPGA based matrix inversion algorithm.

For implementation in a FPGA based environment it is important to optimize the algorithm to reduce complexity and system usage. One method of optimization involves converting the floating point arithmetic into fixed point arithmetic. Determining fixed point values that are suitable as replacements for floating point values can be a difficult and tedious process, with a lot of trial and error. A method has been developed to address this problem for use with the Sherman-Morrison method [17]. The method presented develops a simulation based approach to defining the fixed point values by solving an overdetermined problem using least squares approximations.

Further simplifications can be achieved by reducing the number of multiplication steps required for performing complex multiplication. It has been shown that a complex multiplication can be achieved with as little as 3 multiplications of real numbers, as opposed to 4 multiplications used in previous methods [18]. This will reduce the resources required for the final implementation in the FPGA based hardware. However in the original paper by Sherman-Morrison it was only specified that real values could be used. It is unknown at this point if any work has been done on complex valued matrix inversion [14].

Chapter 3

System Design and Testing

To find an algorithm suitable for use in a hardware based environment several different algorithms were tested. Each algorithm was chosen for a particular advantage or feature that it contained that made it a candidate for hardware implementation. Each candidate algorithm was tested and characterized for its suitability for the desired application on the ASKAP array.

The remainder of this chapter will describe each algorithm and why it was chosen. Finally the test procedures will be outlined with an explanation on how the tests were performed and what their purpose was.

3.1 Algorithms tested

A survey and exploration of several techniques for matrix inversion were performed. Each algorithm was chosen because it showed potential for offering a mathematical method for inversion that was suited to efficient implementation in a FPGA.

The first algorithm was suggested by CSIRO as a potential candidate. It was based on a Sherman-Morrison algorithm which was well suited to performing continual updates on an inverse based on a new input vector.

The second method was a modification of the first method. It proposed the removal of the matrix matrix division through the introduction of a scaling factor.

The third method was not found in literature but was developed from the original research paper by Sherman-Morrison [14]. The approach was fully designed around efficient implementation in an FPGA environment but was tested numerically in MATLAB

For the final method was originally designed to run on a Cray Super computer [15]. As it was already set up for parallel computing it presented as an ideal candidate for testing.

All the algorithms were tested using the approaches set out in the test and evaluation section of the chapter.

3.1.1 Haykin implementation

In Radio Astronomical observations, a telescope user will have a collection of points on the sky that they plan to observe, called pointing's. Each pointing could last a few seconds or up to a hour. The properties of a SOI, such as the SNR and flux density, will change for each pointing. Further, for each pointing there will be a contribution of fast changing RFI sources present in the FOV. For an inversion algorithm where the output calculated previously is updated each cycle this presents a problem. The inversion will treat all data as being equally relevant to the current calculation. This means that as the telescope changes direction, the inversion of the ACM will still include data from a previous pointing. If we are to generate an inverse autocovarience matrix that is relevant to the SOI for a particular moment of time, a forgetting factor term will need to be introduced to the Sherman-Morrison formula, to ensure the more recent data is treated with more importance than older data. To this end it is proposed that the forgetting factor λ be introduced where λ can take a value between 0 and 1 and would typically be around 0.95.

An existing algorithm had been developed by CASS in MATLAB which is an implementation of the Sherman-Morrison-Woodbury formula that included the λ term. A copy of the code can be found in appendix A.1. This algorithm acted as a starting point for the testing and optimization.

The approach used to solve the inverse matrix problem is based off equation 2.8. Eqn 2.8 was extended to include the forgetting factor resulting in eqn 3.1.

$$\mathbf{R}^{-1} = (\mathbf{R}_{i-1}^{-1} + \mathbf{u}_i\mathbf{u}_i^H)^{-1} = \mathbf{R}_{i-1}^{-1} - \frac{(\mathbf{R}_{i-1}^{-1}\mathbf{u}_i\mathbf{u}_i^H\mathbf{R}_{i-1}^{-1})}{1+\mathbf{u}_i^H\mathbf{R}_{i-1}^{-1}\mathbf{u}} \tag{2.11}$$

$$\mathbf{R}^{-1} = \frac{1}{\lambda}\mathbf{R}_{i-1}^{-1} - \frac{1}{\lambda}\frac{(\mathbf{R}_{i-1}^{-1}\mathbf{u}_i\mathbf{u}_i^H\mathbf{R}_{i-1}^{-1})}{1+\mathbf{u}_i^H\mathbf{R}_{i-1}^{-1}\mathbf{u}} \tag{3.1}$$

The following block diagram explains the flow of data and some aspects of the control structure in this algorithm.

The Den unit takes the R^{-1} inverse calculated from the previous cycle. It produces a new

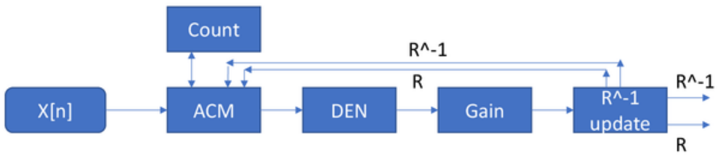


Figure 3.1: Block diagram of the Woodbury implementation developed by CSIRO

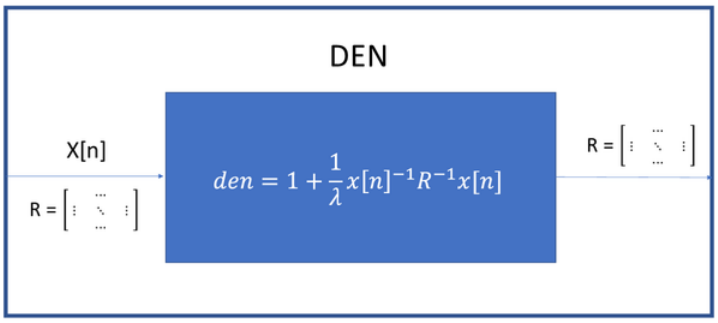


Figure 3.2: Detailed explanation of existing DEN block

matrix, Den. The gain block takes Den and divides R^{-1} by it. This is a matrix matrix division. The updater then brings all the terms together with some adjustments.

Figure 3.1-3.4 show the sections of the existing Sherman-Morrison-Woodbury implementation. The counter is a control process that causes the system to loop N times. The looping of the algorithm is used to test the stability as a function of time. In an continuous system N would loop forever or until the system is reset. To save time with simulations the matrix size was set to be a square matrix of size 20.

This algorithm was hypothesized as being able to perform the update to an inverse without having to re-compute the ACM and then find the difference with the old ACM. Some literature did show that this algorithm could result in a increasing error in the inverse calculated. It was also stated that the error may converge after a number of cycles.

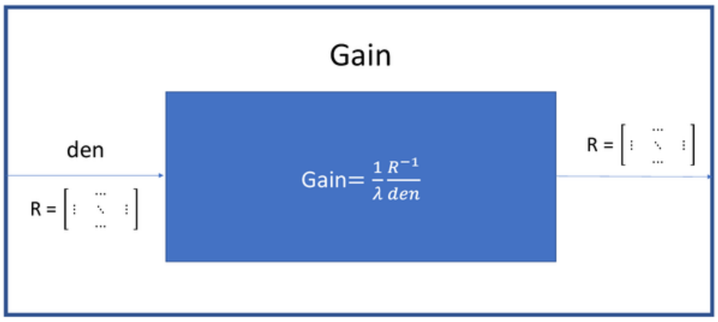


Figure 3.3: Detailed explanation of existing Gain block

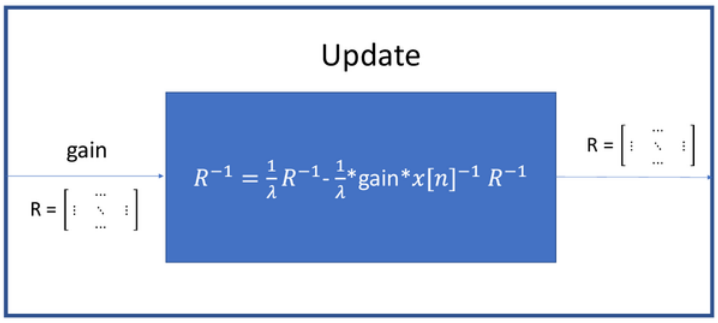

Update
gain
R =
$R^{-1} = \frac{1}{\lambda}R^{-1} - \frac{1}{\lambda}*gain*x[n]^{-1}\ R^{-1}$
R =

Figure 3.4: Detailed explanation of existing update block

To test this algorithm several scripts were developed in MATLAB. To test the response to input data type two data types were generated. The first one was random numbers of double precision floating type. The results of tests with the random numbers were then compared to the results of the algorithm with a band limited source set at an angle to the array.

Using a band limited signals as the input to the matrix inverter the output error was measured. This test was also used to determine if the error converged to a stable point.

3.1.2 Modified Haykin algorithm

Using the results found in eqn 2.13 and eqn 2.14 a novel approach to solving the inverse matrix problem for radio astronomy is presented as a potential solution that is less resource intensive than the method presented in eqn 3.1 [16].

This approach involves the re-arrangement of the equation used in eqn 2.8 in order to remove the division. The removal of the division was achieved by the introduction of a scaling factor α resulting in the equation becoming

$$\alpha_i \mathbf{R}^{-1} = \mathbf{R}_{i-1}^{-1}(\alpha i - 1 + \mathbf{u}_i^H \alpha_{i-1} \mathbf{R}_{i-1}^{-1} \mathbf{u}_i) - (\alpha_{i-1} \mathbf{R}_{i-1}^{-1} \mathbf{u}_i \mathbf{u}_i^H \alpha_{i-1} \mathbf{R}_{i-1}^{-1}) \tag{2.13}$$

$$\alpha_{i+1} = \alpha_i(\alpha_i + \mathbf{u}_{i+1}^H \alpha_i \mathbf{R}_i^{-1} \mathbf{u}_{i+1}) \tag{2.14}$$

Implementation of the algorithm required some modifications to the original design. The algorithm was fully implemented in MATLAB for testing and characterization. The same input data used on the original equation was used to the modified algorithm. The output of the algorithm was observed and compared to the outputs of the original and also to the LU method. The goal of the test was to determine if the new method gives similar results to the original method.

This algorithm was selected for testing as it offered significant advantages due to it reduced complexity of implementation in a hardware based environment. The removal of the division significantly reduced the complexity of the implementation, even though it did result in extra α term being added into the equation.

3.1.3 Sherman-Morrison original method

The paper that first introduced the Sherman-Morrison algorithm proposed a method for performing a rank-1 update of a inverse matrix based on a difference matrix [14]. This approach was originally discounted as it would involve the calculation of an ACM and a difference matrix. It also required 4 matrix of size N, where N is the umber of input elements to a system, to be stored potentially taking up large portions of system memory. This approach was however latter re-introduced as it was discovered that the ASKAP telescope already computes an ACM and this method would therefore be achievable.

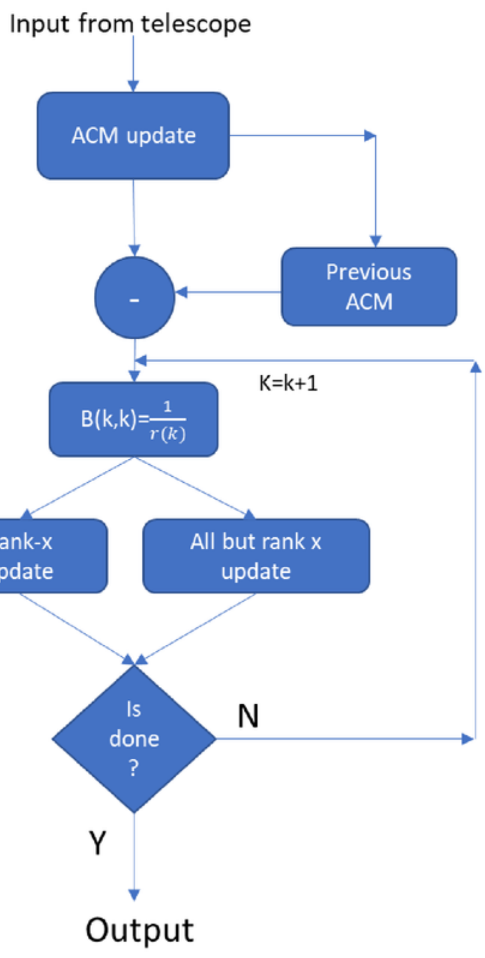


Figure 3.5: Sherman-Morrison inspired algorithm developed

This approach was investigated as it had the potential for large scale efficient implementation in hardware. It does however require one division operation to be performed before any updates can begin. However the division is a single 16 bit number and would be easily achievable in a DSP slice or through bit shifting if it is a quantized number.

3.1.4 Vectorized method

The final method that was explored was an algorithm developed for use of a Cray super computer [15]. This method was chosen as it was design for a parallel architecture and was expected would be suitable for implementation in Hardware. The algorithm is a adaptation of the original method presented by Sherman-Morrison [14]. This particular algorithm was written so that it would handle changes to the ACM larger then a rank-1 change. This provided the opportunity to feed real data into the algorithm without having to condition it.

The algorithm worked by looping through the matrix from the first column through to the last column updating the matrix based on the differences between elements in each column.

3.2 Algorithm testing and evaluation

Several methods were developed to test and evaluate each algorithm or method. For all of the tests developed, the method of evaluating the results involved comparing the output of the inverse of the matrix to a standard method for inverse calculation, such as LU-decomposition, which is a typical industry standard approach along with the QR-decomposition.

3.2.1 Random Vs Real

Some methods tested were designed for use on a stream of continuously generated data from a real world situation. An example of a real world situation would be a radio telescope or mobile phone base station sending data from the receiving array as a vector of numbers, where each number represents the sampled amplitude of a signal at that element at that point in time. Each vector would have a small relation the the vector before it and the vector to come after it. To understand the relation between real signals when compared to randomly generated signals a test was developed to compare the error between the output inverse of both data types to the standard LU-decomposition method.

In order to compare the matrix the maximum and minimum values were taken from the difference matrix. The difference matrix in this test involved comparing the random data to LU data and comparing Real data to LU data. By taking the minimum and maximum difference it is possible to see if the variance of method is effected by the input data. The error was measured after running the method for 100000 input sample vectors. This length was chosen as it represents a quick pointing length. Visual inspections were performed to see if any artifacts could be seen in the output indicating there is a problem or feature of a particular method 3.7.

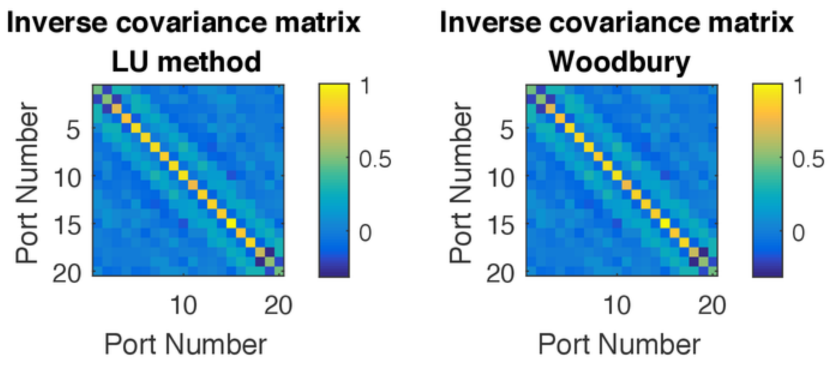


Figure 3.6: The two inverse matrix produced. left: 100000 iteration then LU decomposition, right: 100000 iterations of Woodbury method. Both are normalized

The difference matrix was also able to be visualized as shown in 3.7 and is made by subtracting the two plots in fig 3.6.
The code used to generate the real signal can be seen in appendix A.2.

3.2.2 Error convergence

Another aspect of an algorithms performance is its stability with time. The error of an algorithm will vary over time depending on the input data, rounding and other factors depending on the implementation. An important characteristic of an algorithm will be its ability to remain stable with time. Rather then taking the max and min values for stability with time the Euclidean norms of each matrix were taken and then subtract the value of one norm from the other. This method has been used in previous studies into matrix inversion [19]. An example plot is shown in fig 3.8 showing the error as a function of iterations.

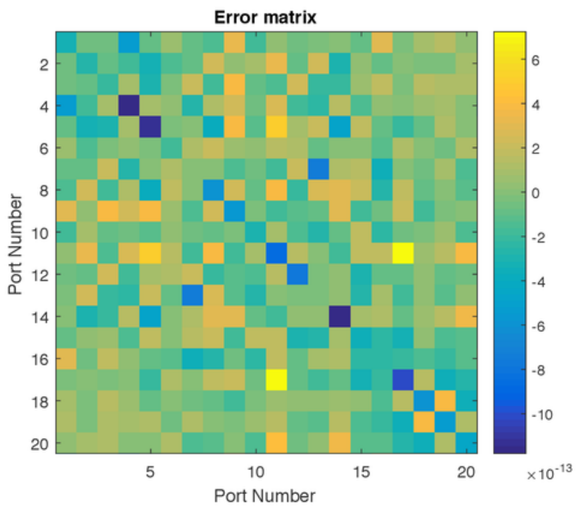


Figure 3.7: Error Matrix of normalized and subtracted gauss and Woodbury methods after 100000 iterations

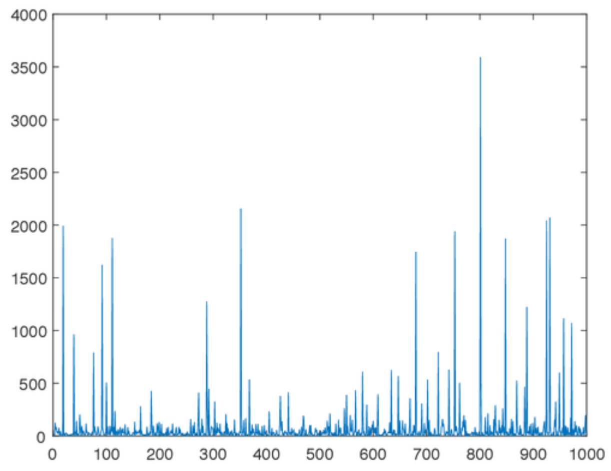


Figure 3.8: Sample of Norms difference as a function of iteration length

3.2.3 SNR dependent error

For an algorithm to correctly invert a matrix of astronomical data correctly it must be immune to variances in the SNR of a signal. To test for this the SNR was varied by dropping the noise power. The angle of the signal source was also tested to determine if there was any change relating to the SOI angle of arrival.

3.2.4 Numerical precision dependent error

Due to the nature of the math operations, when adding or multiplying numbers the resulting number will often be of a larger word size compared to the two original numbers operated on. This can cause significant problems for a hardware implementation of the algorithm. In a hardware based environment all the resources need to be allocated before the program is run. This requires a set size for all the data buses and memory to be decided upon. Further due to limitations of available resources in a fixed hardware system, it is often not practical to allocate enough resources to correctly store the resulting values for numerical operations.

The effects and pitfalls of this problem will be explored in two different areas. The first area will be focusing of the word size and fractional length. For convenience it was decided to use a 16 bit word size as this reflected the word size of the data coming in from the ADC's. In standard Xilinx FPGA hardware there are numerical processing slices refereed to as DSP48. These slices can operate on numbers up to 48 bits for all functions implemented in these slices. This gives an upper maximum for an individual calculation in a DSP48 slice. To fully understand the requirements of the algorithm various points of the algorithm will be probed and the data size and values read to gain a better picture of how it works and what its requirements are. This test will only be performed on the Vector method, but the information gained will be transplantable to any of the algorithms presented.

The second approach will be to test different rounding and enumeration methods. This will be required information for implementation in hardware, as rounding will be required as part of the implementation to ensure numbers do not overflow and cause numerical instability. This will be tested by using a set word, fraction size and testing different options such as convergent, Floor, Zero, Ceiling and Nearest. MATLAB summaries the information of these types based on cost(cpu time), bias and possibility of overflow tab 3.1.

Table 3.1: Rounding types and trade offs credit:mathworks

Mode	Cost	Bias	Possibility of overflow
Ceiling	Low	Large positive	Yes
Convergent	High	Unbiased	Yes
Zero	Low	large for (-) small for (+)	No
Floor	Low	Large negative	No
Nearest	Moderate	Small positive	Yes
Round	High	large for (+) small for (-)	Yes

Chapter 4

Results

This chapter presents the findings form experiments and simulations conducted on the various algorithms presented in the previous chapter. The chapter splits the results into sections based on the type of testing undertaken.

4.1 Effects of Real vs Random Data

In order to correctly evaluate the effectiveness of an algorithm it was important to know what the effect of the different types of input data had on the algorithms.

4.1.1 Real Vs Random

Using the Haykin's algorithm two test scripts were developed in MATLAB (see appendix A.1 and A.2). Both scripts processed the data the same way. The algorithm was set to run for 100000 iterations. The two results were compared showing that the difference was lower for real data when compared to randomly generated data.

Table 4.1: Comparing range of difference matrix between random numbers and band limited signals

Random		Real	
min	max	min	max
-1.19E-15	4.22E-15	-6.34E-13	4.75E-13
-2.4425E-15	1.9984E-15	-8.3100E-13	6.7465E-13
-1.5543E-15	1.9984E-15	-4.6663E-13	4.1867E-13

This experiment was repeated three times to ensure that the data followed the same trend was shown for all ranges. If the trends were not consistent more simulations would have been run to find a more accurate average.

4.1.2 SNR

The effect of SNR was observed and the results tabulated in table 4.2. The error was calculated by comparing the norm to the norm of the LU-method. The signal was set to 0dB and the noise was swept from -10dB down to -80dB. The simulated real band limited signal was used for this test.

Table 4.2: Results from changing the SNR with the SOI set to 0dB

Signal level dB	Noise level dB	Angle of arrival deg	Error
0	-10	70	-8.022E-13
0	-20	70	-1.1147E-12
0	-30	70	-1.17E-10
0	-40	70	-1.16E-10
0	-50	70	-3.57E-9
0	-60	70	-1.125E-7
0	-70	70	-3.468E-6
0	-80	70	-6.69E-5

4.1.3 Angle of Source

To determine if the angle to a SOI from the array face had an effect on the error the angle of the SOI was changed and the error measured. Te results were tabulated in table 4.3. The simulated real band limited signal was used for this test.

Table 4.3: Effect of changing arrival angle on the inversion algorithm

Signal level dB	Noise level dB	Angle of arrival deg	Error
0	-10	70	-1.04E-12
0	-10	60	-7.7E-14
0	-10	50	-3.20E-13
0	-10	40	-6.36E-13
0	-10	30	-7.69E-13
0	-10	20	2.97E-13
0	-10	10	5.92E-13

4.2 Stability with Time

Testing the Haykin's model for stability using random data as in input gave the results plotted in the fig 4.1 below. It can be seen that as the number of iterations increases the error converges to a maximum level. This level remains constance for longer iteration lengths such as 100000

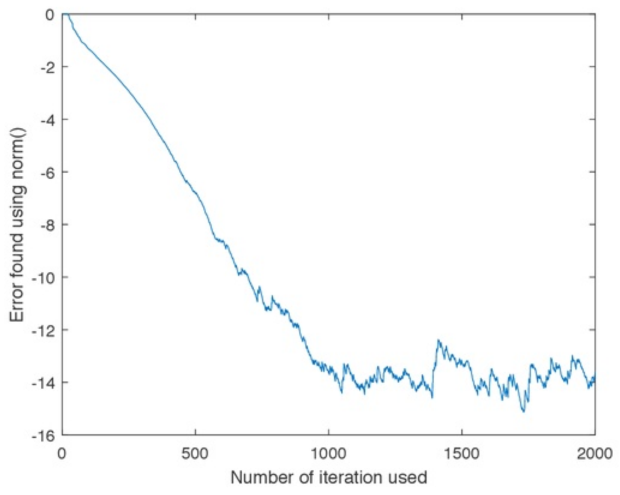


Figure 4.1: The error of the Haykin's model as a function of iterations

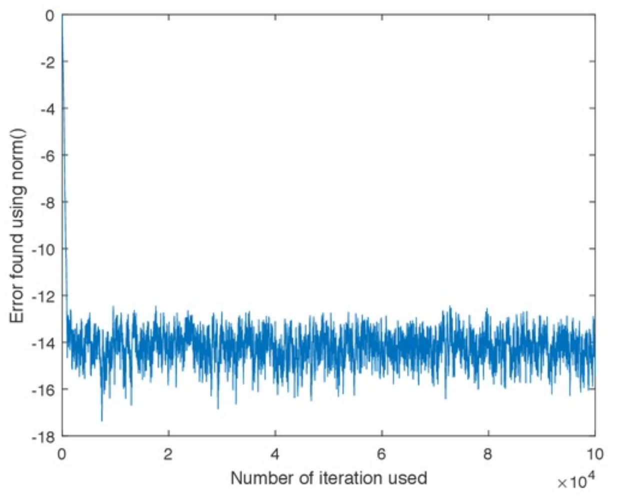


Figure 4.2: The error of the Haykin's model as a function of iterations over a long number of iterations

The modified Haykin's algorithm was tested using the same test script used on the Haykin's algorithm. After several re-designs of the algorithm to implement it in a way that would work mathematically the results in fig 4.3 were generated. It can be seen that this is not a good result.

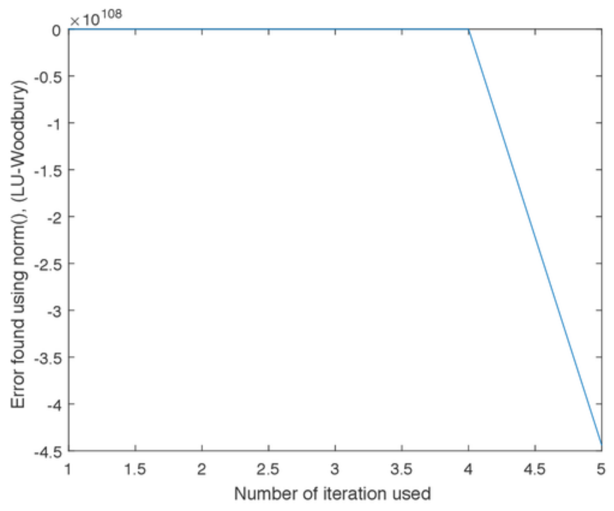


Figure 4.3: The error of the modified Haykin's model as a function of iterations over long number of iterations

After much debugging and following the equation stepping through and monitoring the numbers it was found that the α term would start at 1 and then quickly explode out towards infinity.

4.3 Word Size and Fractional Length

The effects of word size and fractional length were tested on the vectorized method. To start with the script was ran with all numbers set a double precision floating point resulting in fig 4.4. The code for the test can be seen in A.3 and the algorithm in A.5. During the tests on the function as floating point functions were replaced with fixed point functions several numerical properties were noticed.

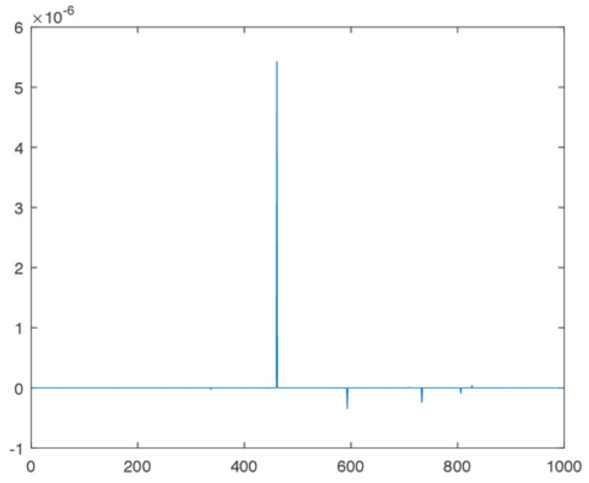


Figure 4.4: The error of double precision Vector method

1. Selection of correct data type(word size and fractional length) is important to reduce error
2. Data types must be consistent in the algorithm or inconsistent results were obtained
3. Correct rounding algorithms were needed or an offset in the numbers was noticed

A 16 bit word size was chosen for testing. This word size was chosen because it was the same size as the data coming from the ADC in the ASKAP telescope. Also 16 bit is a standard size and nicely fits into all the DSP slices in a FPGA. The size could be changed but different word sizes are not explored in this paper.

The fractional length of the data was changed at several points to explore the effects. In the first experiment all the word size and fractional lengths were set to be the same i.e 16,0, for all parts of calculation. This resulted in the following result 4.5. The simulation was run by putting a 20X20 matrix of random 16 bit signed integers into the algorithm and comparing the output to the LU decomposition method.

Several mathematical operators including addition, subtraction and multiplication were replaced with fixed point enabled operators. These operators enabled numbers with two different formats, for instance 16,4 and double float to operate on each other. The operators would then cast the numbers back into the preferred format. The operators also provided the option of handling overflow and rounding. This aspect will be addressed in the next section.

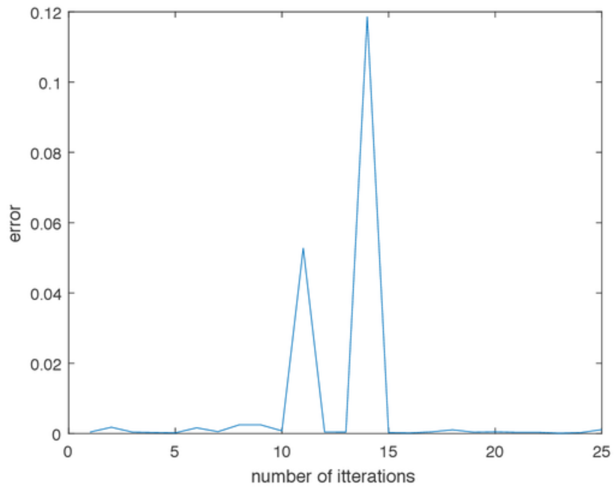


Figure 4.5: The error associated with word/fractional format 16,0

Setting the size to be 16,15 resulted in the error shown in fig 4.6.

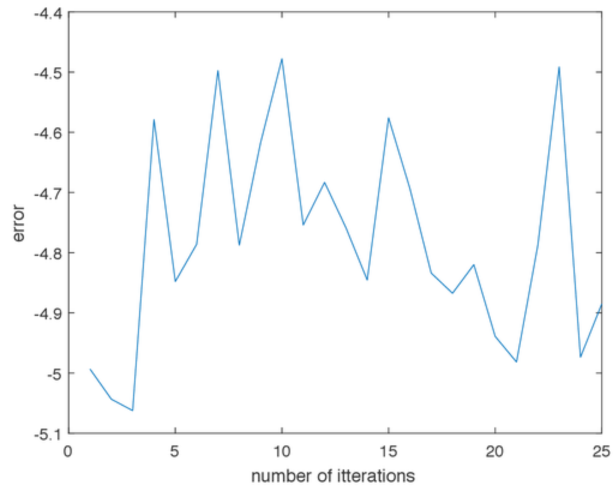


Figure 4.6: The error associated with word/fractional format 16,15

It was hypothesized that there would be some point in between 16,0 and 16,15 that would result in a good solution. However the previous two results showed that 16,0 seamed to be the best configuration. This was not what was expected so the word configuration was sweep from 16,0 to 16,15 resulting in the plot shown in fig 4.7.

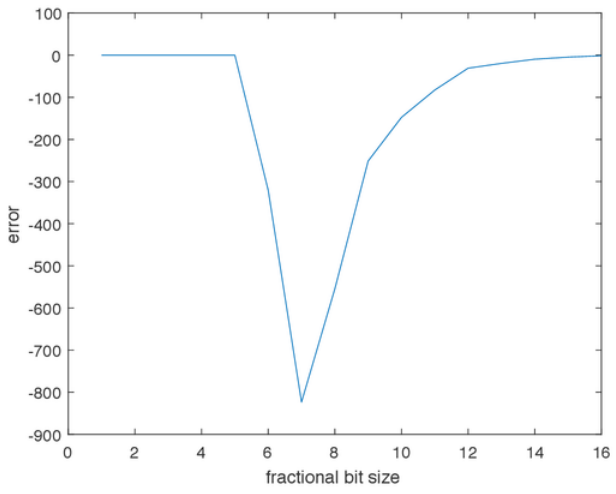


Figure 4.7: The error associated with word/fractional format 16,0 through to 16,15

This results matched with what was hypothesized.

4.4 Rounding

When running the fixed point vectorized method probes were placed at the output of particular operations to observe the size of the resulting numbers. At the output of various multiplications the word size of the numbers was reaching 52 bit. Due to the limitations of 48 bit for a compact implementation in hardware processes for reducing theses numbers were explored. To start with a base line was obtained for a fixed point with no limitation to the word size resulting from an operation. The results can be found in fig 4.8. This result gave a baseline for comparing the effects of number rounding and limiting.

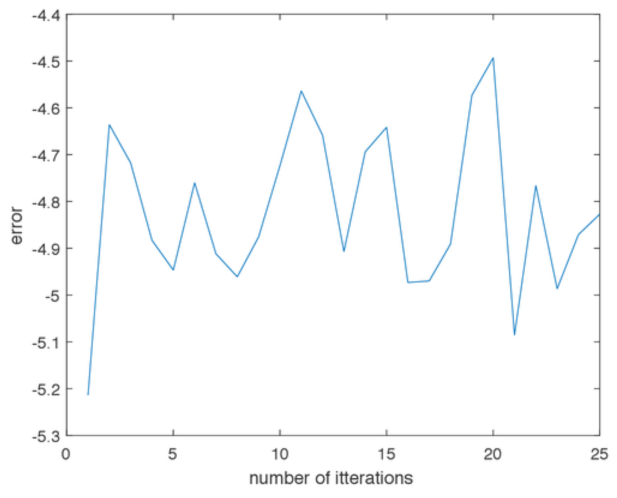


Figure 4.8: The unlimited word size results for the vectorized algorithm. Iterations are along the bottom axis and error is along the vertical

Fixing all operations too 16,15 signed numbers using convergence and wrapping operators resulted the results of fig 4.9.

There appears to be no difference in the error for rounded and no rounded values. this was observed for all types of rounding tested.

4.5 Numerical verification

The algorithm developed from the original paper by Sherman-Morrison was tested using double precision float numbers. The algorithm was able to handle rank-1 updates to the matrix with now errors using the double precision float type. This method was also tested by stepping through the algorithm to ensure that each element in the array was not effected by others during the update procedure. It was found that this method while only handling rank-1 updates, it could be implemented in parallel so each element in the matrix was updated at the same time. To do rank-2 or higher updates successive runs of the algorithm were required.

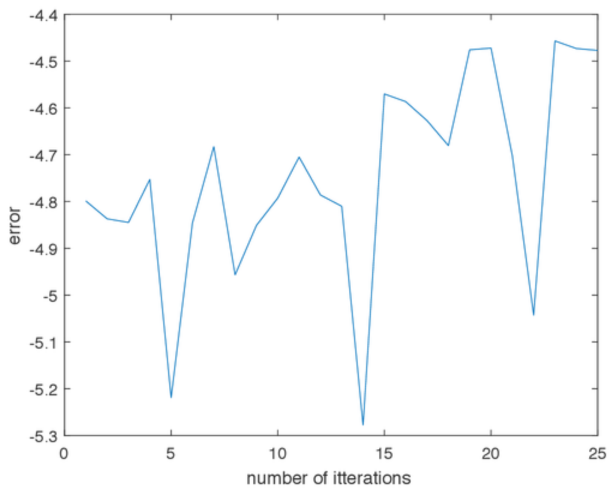


Figure 4.9: The limited word size results for the vectorized algorithm using convergent and wrapping for rounding

Chapter 5

Discussion

During the course of this project several observations were made on the behavior of the various algorithms. In this section these observations will be discussed.

During investigation into the operation of the Haykin method it was noticed that it operated at a matrix level rather then matrix element level. This can be seen in eqn 2.12 where the R^{-1} terms, the U and U^H terms multiply or divide.

$$\mathbf{R}^{-1} = (\mathbf{R}_{i-1}^{-1} + \mathbf{u}_i\mathbf{u}_i^H)^{-1} = \mathbf{R}_{i-1}^{-1} - \frac{(\mathbf{R}_{i-1}^{-1}\mathbf{u}_i\mathbf{u}_i^H\mathbf{R}_{i-1}^{-1})}{1 + \mathbf{u}_i^H\mathbf{R}_{i-1}^{-1}\mathbf{u}} \tag{2.12}$$

This led to some investigation into the operation and effectiveness of this approach. Two conclusions were drawn. Either this is the way the algorithm operates in which case a work around needed to be developed, as performing a matrix division requires the inverse of the matrix to be calculated making a circular problem. Alternatively the method that this algorithm uses had been misinterpreted. The second option was the most likely however was also the most problematic.

Further testing on an algorithm that was developed to overcome the short fall associated with the division proved to produce errors in its output 4.3. This re-enforced the idea that a misinterpretation of the process had occurred. It was found that the terms in the α and R^{-1} matrix's were trending towards infinity.

Rather than spend an unknown amount of time trying to determine where the misinterpretation was, a new approach was decided upon. The new approach involved going back to the original paper and developing an algorithm from scratch. This was achieved with results showing the algorithm gave better results than first thought. The algorithm however could only handle a rank-1 update. There was an approach developed that performed up to rank-k, where k is size of the square matrix, updates by successive repeats of the rank-1 update [15].

It was found through much experiment and testing that both the algorithm developed from the original paper and the expanded version that allowed for updates for $k > 1$ provided stable solutions with little or no error. It was also identified with some modifications a algorithm could be written that to detect only the changes above a set threshold level and perform an update on those parts. This would reduce the problem from a rank-k to a rank-$(1 < x < k)$ problem.

Testing the effects of rounding showed no differences between the different types of rounding. This could be the fractional part of the number having to small of a value to be noticeable on the test scale that were used on these experiments. With longer run times like those found on a telescope a bias might become noticeable in those areas.

Chapter 6

Conclusions

In order to solve the matrix inversion problem a algorithm has been developed that builds on the work presented in the original Sherman-Morrison paper on matrix inversion. It is proposed that an algorithm using this approach with a modified update filter would effectively solve the inversion problem. It was found through testing that the rank-1 method proposed by Sherman-Morrison was in fact the most efficient algorithm for matrix update for a rank-1 update. Further it was also shown to be the most accurate if appropriate selection of word and fractional size was used. This approach is well suited to implementation in a FPGA based environment.

To over come the limit of rank-1 updates a second algorithm was tested that assumed a rank-k update is always required, where k is the size of square matrix. This method proved to have slightly higher error level, however the overall error was much lower when compared to other methods tested. This method did also require the ACM to be computed and a difference matrix generated. This initially seamed to be a draw back of this method however the ACM update was found to be performed as part of the other algorithms. This points to comparable execution time for the two different approaches.

It is believed with an appropriate filter on the update matrix the overall execution time could be reduced further for this algorithm. This filter would employ a threshold based detector and would only perform updates to the inverse for elements in the ACM for which a change higher then the threshold have occurred. This would remove unnecessary calculations from being performed.

It was found through numerical and simulation based investigation that the both the Haykin and modified Haykin algorithms are not suitable for implementation in a hardware based environment such as a FPGA. This was due to performing division of matrix operators and several terms trending towards infinity.

Real data was shown to have an effect on the Haykin algorithm offering lower error level when compared to randomly generated data. With use of real data the SNR was observed to have an effect. A higher SNR value will result in a lower error. Angle of signal arrival at an array was found to have no effect which is what would be expected for a PAF. Random data was used for the remaining algorithms as they were developed to handle random data.

The method developed from the Sherman-Morrison paper was shown to remain stable with time. The Haykin's method however did don remain stable for the first 2000 iteration buy converged to a point in a roughly linear fashion. After 2000 iterations it was shown to be stable but had a large error. The modified Haykin's method showed no stability but was found to tend toward infinity within a few iterations.

Word size and fractional length were shown to play a major role in minimizing error. Moving to fixed point numbers did result in an error increases in all algorithms. For the Sherman-Morrison and Vectorized algorithms this error was able to be controlled through correct selection of fractional length. The type of rounding used for converting data types has not been measured as having an effect however there could still be a aver error introduced that was not found with the short runs used to test all the different methods.

It was also found during the course of this project that no research has been conducted into performing inversion updates for complex numbers. It is therefore unknown at this point in time if this algorithm could be applied to a matrix of complex numbers.

Chapter 7

Future work

This project was part of a much larger project with the goal of implementing a near real time adaptive beam former for the CSIRO PAF's on the ASKAP array in the MRO. In order to implement an adaptive beam former a method for determining beam weights was required. Several potential methods were identified with one of the more promising methods using matrix inversion updates to perform a fast update of the Eigenvalue problem [20]. The findings of this project have demonstrated a method for performing the update of the matrix inversion. Using the updated inverse it is now possible, with some more research, to implement an Eigenvalue update.

With a method of performing a matrix inverse up date presented and a simple algorithm for performing a Eigenvector update found in literature it is now possible to start development on a fast, hardware based beam former. This beam former is potentially capable of performing real time null point steering of a PAF. This makes it now possible to implement real time RFI mitigation techniques that have previously been impractical.

The next step in this line of research comes in two parts. The first part will involve further research and optimization of the Eigenvalue problem. The approach developed in this project would be highly applicable to solving the Eigenvector problem and it is recommended that the methods are considered for future use.

The second progression form this project will be to target the presented findings to a hardware based system. This is a critical step in ensuring the methods developed in this project are simulating real word applications correctly.

It is believed that with further research and implementation of the findings of this project and future similar projects it will be possible to implement a real time active RFI mitigation beam former for techniques such as null point steering. This would result in a better SNR in the signals from the beam former entering the correlator. This would reduce the required integration time on the telescope which in tern will increase the amount of science that is capable of being produced. The amount of time that astronomers lose to blocking out bad data from RFI sources would also be significantly reduce resulting in astronomers being able to achieve more productive goals.

As with all new technology it will take time to demonstrate to the astronomical community that these techniques will result in correct data and are worth the time and effort to implement. Several more in-depth research projects will need to be carried out to independently verify the proposed algorithm presented in conclusion of this project.

Further research could also explore aspects of the algorithm associated with the processing of complex numbers.

The next phase of work for the authors of this paper will be to implement the proposed algorithm in an FPGA environment. The target platform with be either the Xilinx Zynq or Artix 7 platforms.

Chapter 8

Abbreviations

AMBA	Advanced Microcontroller Bus Architecture
ARM	Acorn RISC Machine/Advanced RISC Machine
ASKAP	Australian Square Kilometer Pathfinder Project
AXI	Advanced eXtensible Interface protocol
CASS	CSIRO Astronomy and Space Science
CSIRO	Commonwealth Scientific and Industrial Research Origination
DDR	Double Data Rate
DSP	Digital Signal Processing
FOV	Field Of View
FPGA	Field Programmable Gate Array
JTAG	Joint Test Action Group
LMS	Least Mean Square
LNA	Low Noise Amplifier
PAF	Phased Array Feed
PL	Programmable Logic
PS	Processor Segment
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
RF	Radio Frequency
SKA	Square Kilometer Array
SOC	System On Chip
SOI	Signal Of Interest

Appendix A

MATLAB scripts

A.1 Haykin method 1

The following file was the file provided by CASS as a starting point for the project. It produces a covariance matrix and inverse of the covariance matrix through LU-decomposition and through the use of the Haynkin Woodbury implementation

```
K = 20; % measurement vector length
N = 100000; % number of iterations to run the update for
lambda = 0.995; % forgetting factor for the recursive update

R = eye(K);
inv_lambda = 1/lambda;
R_inv = eye(K); % initialise the inverse matrix to be updated
for lp = 1:N
    xvec = randn(K,1); % new measurement vector
     R = lambda.*R + (1-lambda).*(xvec*xvec'); % update the covariance matrix

    % Use Woodbury's identity to update the inverse covariance matrix
     den = 1 + inv_lambda*xvec'*R_inv*xvec;
     gain_vec = inv_lambda*R_inv*xvec./den;
     R_inv = inv_lambda*R_inv - inv_lambda*gain_vec*xvec'*R_inv;
end

R_inv_calc = inv(R);

% normalise to the largest element to make comparison by subtraction
% possible
R_inv = R_inv/max(max(R_inv));
R_inv_calc = R_inv_calc/max(max(R_inv_calc));

% compute the difference matrix (error matrix)
diff_inv = R_inv - R_inv_calc;
disp(max(max(diff_inv)))
disp(min(min(diff_inv)))
```

```
figure(3)
clf
subplot(1,2,1)
imagesc(R_inv)
colorbar
axis square
title({'Inverse covariance matrix','LU method'})
xlabel('Port Number') % x-axis label
ylabel('Port Number') % y-axis label
subplot(1,2,2)
imagesc(R_inv_calc)
axis square
colorbar
title({'Inverse covariance matrix','Woodbury'})
xlabel('Port Number') % x-axis label
ylabel('Port Number') % y-axis label

figure(4)
clf
imagesc(diff_inv)
axis square
colorbar
title('Error matrix')
xlabel('Port Number') % x-axis label
ylabel('Port Number') % y-axis label
```

A.2 Haykin method 2

This program modified the original Haykin implementation to include a signal source rather than random numbers. The signal source was a band limited signal produced from the simulated response of 20 receiving nodes on a linear array.

```
%% Initialise variables
K = 20; % measurement vector length (number of array elements)
N = 100000; % number of iterations to run the update for
lambda = 0.995; % forgetting factor for the recursive update
fc = 1.0; %centre frequency
Fs = 4*fc; % sampling frequency
omega = 2.0*pi*fc; % define the angular frequency
vel = 1.0; % propagation velocity
sdir = 70; % source direction (relative to array end-fire)
sp = 0.0;          % source power
np = -10.0;        % element self noise

%% Array set-up
% Linear Array
org = [0.0 0.0];                % array phase centre
spac = vel/(2*fc);              % element spacing (half wavelength)
x = spac*(-(K-1)/2:(K-1)/2);    % X-coordinates of the elements
y = zeros(1,K);                 % Y-coordinates of the elements
sensor_pos = [x' y'];

figure(1)
clf
plot(sensor_pos(:,1),sensor_pos(:,2),'.','MarkerSize',20)
axis square
grid on

%% Generate the element voltage signals
% Convert source powers to W
sp = 10^(sp/10);
np = 10^(np/10);

% Compute the sensor delay vector for the given source direction
aoa = sdir/180*pi; % angle-of-arrival
source_vec = [cos(aoa) sin(aoa)];
% tau = (x*cos(angle) + y*sin(angle))/vel;
tau = (sensor_pos*source_vec')/vel;

% Simulate a band-limited source
Ns = 100; %number of sinusoids
t = (0:N-1)*1/Fs;
src_phs = rand(1,Ns) * 2*pi; % generate a vector of random phases between 0 and 2Pi
src_freq = linspace(fc*0.9,fc*1.1,Ns);
src = zeros(K,N);
for element = 1:K
```

```
    for lp = 1:Ns
        src(element,:) = src(element,:) + sqrt(sp)*cos(2*pi*src_freq(lp)*(t+tau(element)) + s
    end
end
% add element self-noise
meas_mat = src + sqrt(np)*randn(K,N);

pwr_spect_1 = 20*log10(abs(fftshift(fft(meas_mat(1,:)))));
figure(2)
clf
plot(pwr_spect_1(1:N/2-1))

%% compute the array covariance matrix
R = eye(K);
inv_lambda = 1/lambda;
R_inv = eye(K); % initialise the inverse matrix to be updated
for lp = 1:N
    xvec = meas_mat(:,lp); % new measurement vector
    R = lambda.*R + (1-lambda).*(xvec*xvec'); % update the covariance matrix

    % Use Woodbury's identity to update the inverse covariance matrix
    den = 1 + inv_lambda*xvec'*R_inv*xvec;
    gain_vec = inv_lambda*R_inv*xvec./den;
    R_inv = inv_lambda*R_inv - inv_lambda*gain_vec*xvec'*R_inv;
end

R_inv_calc = inv(R);

% normalise to the largest element to make comparison by subtraction
% possible
R_inv = R_inv/max(max(R_inv));
R_inv_calc = R_inv_calc/max(max(R_inv_calc));

% compute the difference matrix (error matrix)
diff_inv = R_inv - R_inv_calc;

figure(3)
clf
subplot(1,2,1)
imagesc(R_inv)
colorbar
axis square
title({'Inverse covariance matrix','LU method'})
xlabel('Port Number') % x-axis label
ylabel('Port Number') % y-axis label
subplot(1,2,2)
imagesc(R_inv_calc)
axis square
colorbar
title({'Inverse covariance matrix','Woodbury'})
xlabel('Port Number') % x-axis label
ylabel('Port Number') % y-axis label
```

```
figure(4)
clf
imagesc(diff_inv)
axis square
colorbar
title('Error matrix')
xlabel('Port Number') % x-axis label
ylabel('Port Number') % y-axis label
```

A.3 Iteration error measurement

This script was used to measure the error from the Haynkin method as a function of the number of iterations.

```
%itteration vs error test script
%% set-up
N=100000; %maximum number of iterations for update
results=[]; %error value stored in here
K = 20; % measurement vector length (number of array elements)
lambda = 0.995; % forgetting factor for the recursive update
fc = 1.0; %centre frequency
Fs = 4*fc; % sampling frequency
omega = 2.0*pi*fc; % define the angular frequency
vel = 1.0; % propagation velocity
sdir = 70; % source direction (relative to array end-fire)
sp = 0.0;            % source power
np = -10.0;          % element self noise
errorvalue=zeros(1,N);

%% Array set-up
% Linear Array
org = [0.0 0.0];                % array phase centre
spac = vel/(2*fc);              % element spacing (half wavelength)
x = spac*(-(K-1)/2:(K-1)/2);    % X-coordinates of the elements
y = zeros(1,K);                 % Y-coordinates of the elements
sensor_pos = [x' y'];

figure(1)
clf
plot(sensor_pos(:,1),sensor_pos(:,2),'.','MarkerSize',20)
axis square
grid on
%% Generate the element voltage signals
% Convert source powers to W
sp = 10^(sp/10);
np = 10^(np/10);

% Compute the sensor delay vector for the given source direction
aoa = sdir/180*pi; % angle-of-arrival
source_vec = [cos(aoa) sin(aoa)];
% tau = (x*cos(angle) + y*sin(angle))/vel;
tau = (sensor_pos*source_vec')/vel;

% Simulate a band-limited source
Ns = 100; %number of sinusoids
t = (0:N-1)*1/Fs;
src_phs = rand(1,Ns) * 2*pi; % generate a vector of random phases between 0 and 2Pi
src_freq = linspace(fc*0.9,fc*1.1,Ns);
src = zeros(K,N);
```

```
for element = 1:K
    for lp = 1:Ns
        src(element,:) = src(element,:) + sqrt(sp)*cos(2*pi*src_freq(lp)*(t+tau(ele
    end
end
% add element self-noise
meas_mat = src + sqrt(np)*randn(K,N);

pwr_spect_1 = 20*log10(abs(fftshift(fft(meas_mat(1,:)))));
figure(2)
clf
plot(pwr_spect_1(1:N/2-1))

%% compute the array covariance matrix
R = eye(K);
inv_lambda = 1/lambda;
R_inv = eye(K); % initialise the inverse matrix to be updated
for lp = 1:N
    xvec = meas_mat(:,lp); % new measurement vector
    R = lambda.*R + (1-lambda).*(xvec*xvec'); % update the covariance matrix

    % Use Woodbury's identity to update the inverse covariance matrix
    den = 1 + inv_lambda*xvec'*R_inv*xvec;
    gain_vec = inv_lambda*R_inv*xvec./den;
    R_inv = inv_lambda*R_inv - inv_lambda*gain_vec*xvec'*R_inv;
    R_inv_calc = inv(R);

%     % normalise to the largest element to make comparison by subtraction
%     % possible
%     R_inv = R_inv/max(max(R_inv));
%     R_inv_calc = R_inv_calc/max(max(R_inv_calc));

    % compute the difference matrix (error matrix)
    errorvalue(lp) = norm(R_inv) - norm(R_inv_calc);
end

disp('Done');
figure(5)
plot(errorvalue);
xlabel('Number of iteration used') % x-axis label
ylabel('Error found using norm()') % y-axis label
```

A.4 Vector Method First test script

This script was the first script used to test the vectorized method. latter this was replaced with a version that was able to handle fixed point arithmetic.

```
%%Sherman-Morrsions Tester 01
%tests the algorythims to be implemented in HDL

%%setup
clear
N=20;   %matrix size
n=1000;   %number of itterations

D=zeros(n,1);
%%test run setup
%first pass for lu method fixed point
for i=1:n
    A=rand(N);
    %disp('Original A');

    %disp('LU-method B');
    C=inv(A);

    %%Testing
    B=eye(N);
    B=vectorInvert01(A, B,N);
    %disp('fixed sherman-morrison method B');
    D(i)=norm(C)-norm(B);
    %disp('diffrence matrix');
end
plot(D)
disp(sum(D)/n)
```

A.5 Vector Method Second test script

This script represents the vectorized method. This is the last version of the script and was used to verify the changes to accuracy as a function of fraction size.

```
%%Sherman-Morrsions Tester 01
%tests the algorythims to be implemented in HDL

%%setup
clear
N=20;    %matrix size
n=25;    %number of itterations
F = fimath('OverflowAction','Saturate','RoundingMethod','Convergent');

frac=15;
D=zeros(n,1);
%%test run setup
%first pass for lu method fixed point
parfor i=1:n
    %frac=frac+1;
    datatype1=fi([],1,16,0);
    datatype2=fi([],1,16,frac);
    disp(i)
    A=randi([-32768,32767],N);
    %disp('Original A');

    %disp('LU-method B');
    C=inv(A);

    %%Testing convergent
    A=cast(A,'like',datatype1);
    B1=eye(N,'like',datatype2);
    B1=vectorInvert06(A, B1,N,F,frac);
    B1=double(B1);
    %disp('fixed sherman-morrison method B');
    D1(i)=norm(C)-norm(B1);
    %disp('diffrence matrix');
end
plot(D1)
hold on
xlabel('number of itterations') % x-axis label
ylabel('error') % y-axis label
disp(sum(D)/n)
```

A.6 Vector Algorithm 1

This was the original script first script taken from the previous research [15].

```
function [B] = vectorInvert01( A, B,n)
%UNTITLED cell by cell sherman-morrison matrix inversion
%   Detailed explanation goes here
%%setup

%%Pre-calculation
V=A-B;

%%Calculation
for k=1:n
    r(k)=1+V(k,:)*B(:,k);
    B(k,k)=1/r(k);
    for i=1:k-1
        B(i,k)=B(k,k)*B(i,k);
    end
    for j=1:k-1
        esc=-V(k,:)*B(:,j);
        for i=1:k
            B(i,j)=B(i,j)+esc*B(i,k);
        end
    end
    for j=k+1:n
        esc=-V(k,:)*B(:,j);
        for i=1:k
            B(i,j)=B(i,j)+esc*B(i,k);
        end
    end
end


end
```

A.7 Vector Algorithm 5

This script was the final form of the algorithm that had been optimized for implementation in a FPGA environment.

```
function [B] = vectorInvert05( A, B,n, F,frac)
%UNTITLED cell by cell sherman-morrison matrix inversion
%   converted for fixed point operations
%%setup

%%Pre-calculation
V=accumneg(A,B,'Convergent','wrap');
neg1=fi(-1,1,16,frac);
%%Calculation
for k=1:n
    a=fi(0,1,16,frac);
    for i=1:n
        tmp=mpy(F,V(k,i),B(i,k));
        %a=a+tmp;
        a=accumpos(a,tmp,'Convergent','wrap');
    end
    tmp=fi(1,1,16,frac);
    disp('tmp')
    disp(tmp)
    disp('a')
    disp(a)
    r=accumpos(tmp,a,'Convergent','wrap');
    disp('r')
    disp(r)
    s=quantize(r,1,16,frac);
    disp('s')
    disp(s)
    if s~=0
        B(k,k)=1/s;
    end
    for i=1:k-1
        B(i,k)=mpy(F,B(k,k),B(i,k));
    end
    for j=1:k-1
        esc=0;
        for i=1:n
            tmp=mpy(F,V(k,i),B(i,j));
            tmp=mpy(F,neg1,tmp);
            esc=fi(accumpos(esc,tmp,'Convergent','wrap'),1,16,frac);
        end
        for i=1:k
            B(i,j)=accumpos(B(i,j),mpy(F,esc,B(i,k)),'Convergent','wrap');
        end
    end
    for j=k+1:n
```

```
        esc=0;
        for i=1:n
            tmp=mpy(F,V(k,i),B(i,j));
            tmp=mpy(F,neg1,tmp);
            esc=fi(accumpos(esc,tmp,'Convergent','wrap'),1,16,frac);
        end
        for i=1:k
            B(i,j)=accumpos(B(i,j),mpy(F,esc,B(i,k)),'Convergent','wrap');
        end
    end
end

end
```

A.8 Modified Haykin method

This script was for the modified Haykin method.

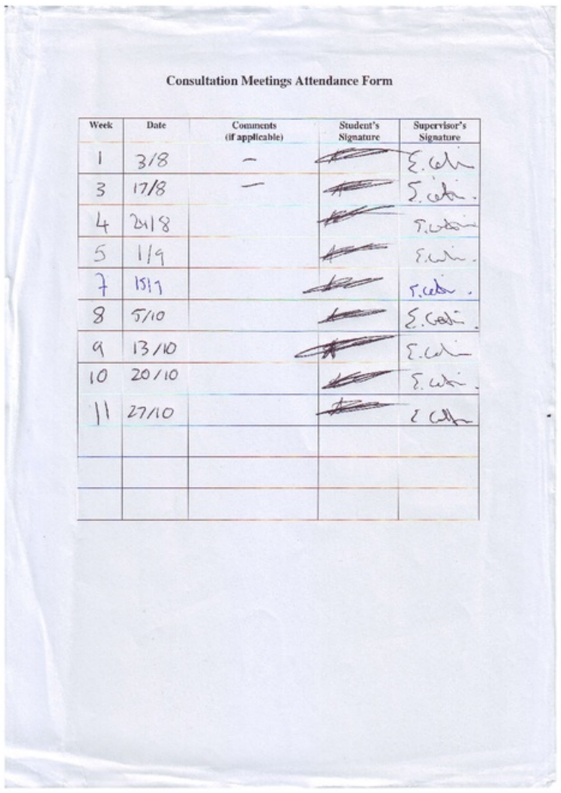
```
%Author:Aaron Sanders
%Date:5/9/17
%Purpose: To test a given function for the error associated with its
%calculation of the matrix inverse. It will compare to the typical inv()
%function built into matlab based off the LU-Decompisition

%% Setup
clearvars
N=1000; %maximum number of itterations for update
%results=[]; %error value stored in here
K = 20; % measurement vector length (number of array elements)
lambda = 0.995; % forgetting factor for the recursive update
fc = 1.0; %centre frequency
Fs = 4*fc; % sampling frequency
omega = 2.0*pi*fc; % define the angular frequency
vel = 1.0; % propagation velocity
sdir = 70; % source direction (relative to array end-fire)
sp = 0.0;           % source power
np = -10.0;         % element self noise
errorvalue=zeros(1,N);

%% Array set-up
% Linear Array
org = [0.0 0.0];
% array phase centre
spac = vel/(2*fc);              % element spacing (half wavelength)
x = spac*(-(K-1)/2:(K-1)/2);    % X-coordinates of the elements
y = zeros(1,K);                 % Y-coordinates of the elements
sensor_pos = [x' y'];

figure(1)
clf
plot(sensor_pos(:,1),sensor_pos(:,2),'.','MarkerSize',20)
axis square
grid on

%% Generate the element voltage signals
% Convert source powers to W
sp = 10^(sp/10);
np = 10^(np/10);

% Compute the sensor delay vector for the given source direction
aoa = sdir/180*pi; % angle-of-arrival
source_vec = [cos(aoa) sin(aoa)];
% tau = (x*cos(angle) + y*sin(angle))/vel;
tau = (sensor_pos*source_vec')/vel;
```

```
% Simulate a band-limited source
Ns = 100; %number of sinusoids
t = (0:N-1)*1/Fs;
src_phs = rand(1,Ns) * 2*pi; % generate a vector of random phases between 0 and 2Pi
src_freq = linspace(fc*0.9,fc*1.1,Ns);
src = zeros(K,N);
for element = 1:K
    for lp = 1:Ns
        src(element,:) = src(element,:) + sqrt(sp)*cos(2*pi*src_freq(lp)*(t+tau(element)) + s
    end
end
% add element self-noise
meas_mat = src + sqrt(np)*randn(K,N);

pwr_spect_1 = 20*log10(abs(fftshift(fft(meas_mat(1,:)))));
figure(2)
clf
plot(pwr_spect_1(1:N/2-1))

%% compute the array covariance matrix
R = eye(K);
R_inv = eye(K); % initialise the inverse matrix to be updated
a(1:K,1:K)=1;
xvec = meas_mat;
for lp = 2:(N-1)
   R = R + (xvec*xvec'); % update the covariance matrix
   R_inv_calc=inv(R);
   R_inv(:,:,lp)=R_inv(:,:,lp-1)*(a(:,:,lp-1)+xvec(:,lp)'*R_inv(:,:,lp-1)*xvec(:,lp-1))-(R_i
   a(:,:,lp)=a(:,:,lp-1)*(a(:,:,lp-1)+xvec(:,lp+1)'*a(:,:,lp-1)*R_inv(:,:,lp)*xvec(:,lp+1));


   errorvalue_test1(lp)=norm(R_inv_calc)- norm(R_inv(:,:,lp));
end

R = eye(K);
inv_lambda = 1/lambda;
R_inv = eye(K); % initialise the inverse matrix to be updated
for lp = 1:(N)


    xvec = meas_mat(:,lp); % new measurement vector
    R = lambda.*R + (1-lambda).*(xvec*xvec'); % update the covariance matrix

    % Use Woodbury's identity to update the inverse covariance matrix
    den = 1 + inv_lambda*xvec'*R_inv*xvec;
    gain_vec = inv_lambda*R_inv*xvec./den;
    R_inv = inv_lambda*R_inv - inv_lambda*gain_vec*xvec'*R_inv;
    errorvalue_CASS(lp) =norm(R_inv_calc)- norm(R_inv);
end
```

```
disp('Done');
figure(5)
plot(errorvalue_test1);
hold on
%plot(errorvalue_CASS);
hold off
xlabel('Number of iteration used') % x-axis label
ylabel('Error found using norm(), (LU-Woodbury)') % y-axis label
```

Appendix B

Records of meetings



Consultation Meetings Attendance Form

Week	Date	Comments (if applicable)	Student's Signature	Supervisor's Signature
1	3/8	–	[illegible]	[illegible]
3	17/8	–	[illegible]	[illegible]
4	24/8		[illegible]	[illegible]
5	1/9		[illegible]	[illegible]
7	15/9		[illegible]	[illegible]
8	5/10		[illegible]	[illegible]
9	13/10		[illegible]	[illegible]
10	20/10		[illegible]	[illegible]
11	27/10		[illegible]	[illegible]

Figure B.1: scan of document of attendance for ENGG411 meetings

References

[1] J. Landon, M. Elmer, J. Waldron, D. Jones, A. Stemmons, B. D. Jeffs, K. F. Warnick, J. R. Fisher, and R. D. Norrod, “Phased array feed calibration, beamforming, and imaging,” *The Astronomical Journal*, vol. 139, no. 3, p. 1154, 2010. [Online]. Available: http://stacks.iop.org/1538-3881/139/i=3/a=1154

[2] G. Moorey, R. Bolton, R. Gough, and H. Kanoniuk, “A 77 - 117 ghz cryogenically cooled receiver for radioastronomy,” 08 2017.

[3] K. F. Warnick, “Interference in radio astronomy and rfi mitigation techniques,” Online presentation, 04 2014. [Online]. Available: http://www.atnf.csiro.au/people/Tasso.Tzioumis/sms2014/presentations/Warnick(RFI_Mitigation).pdf

[4] A. R. Thompson, J. M. Moran, and G. W. S. Jr., *Interferometry and Synthesis in Radio Astronomy*. Springer-Verlag GmbH, 2017. [Online]. Available: http://www.ebook.de/de/product/26469782/a_richard_thompson_james_m_moran_george_w_swenson_jr_interferometry_and_synthesis_in_radio_astronomy.html

[5] L. Staveley-Smith, W. Wilson, T. Bird, M. Disney, R. Ekers, K. Freeman, R. Haynes, M. Sinclair, R. Vaile, R. Webster, and A. Wright, “The parkes 21 cm multibeam receiver,” vol. 13, p. 243, 10 1996.

[6] W. E. Wilson, R. H. Ferris, P. Axtens, A. Brown, E. Davis, G. Hampson, M. Leach, P. Roberts, S. Saunders, B. S. Koribalski, J. L. Caswell, E. Lenc, J. Stevens, M. A. Voronkov, M. H. Wieringa, K. Brooks, P. G. Edwards, R. D. Ekers, B. Emonts, L. Hindson, S. Johnston, S. T. Maddison, E. K. Mahony, S. S. Malu, M. Massardi, M. Y. Mao, D. McConnell, R. P. Norris, D. Schnitzeler, R. Subrahmanyan, J. S. Urquhart, M. A. Thompson, and R. M. Wark, “The australia telescope compact array broadband backend (cabb).”

[7] W. Gawronski, “Control and pointing challenges of large antennas and telescopes,” *IEEE Transactions on Control Systems Technology*, vol. 15, no. 2, pp. 276–289, March 2007.

[8] A. W. Hotan, J. D. Bunton, L. Harvey-Smith, B. Humphreys, B. D. Jeffs, T. Shimwell, J. Tuthill, M. Voronkov, G. Allen, S. Amy, K. Ardern, P. Axtens, L. Ball, K. Bannister, S. Barker, T. Bateman, R. Beresford, D. Bock, R. Bolton,

M. Bowen, B. Boyle, R. Braun, S. Broadhurst, D. Brodrick, K. Brooks, M. Brothers, A. Brown, C. Cantrall, G. Carrad, J. Chapman, W. Cheng, A. Chippendale, Y. Chung, F. Cooray, T. Cornwell, E. Davis, L. de Souza, D. DeBoer, P. Diamond, P. Edwards, R. Ekers, I. Feain, D. Ferris, R. Forsyth, R. Gough, A. Grancea, N. Gupta, J. C. Guzman, G. Hampson, C. Haskins, S. Hay, D. Hayman, S. Hoyle, C. Jacka, C. Jackson, S. Jackson, K. Jeganathan, S. Johnston, J. Joseph, R. Kendall, M. Kesteven, D. Kiraly, B. Koribalski, M. Leach, E. Lenc, E. Lensson, L. Li, S. Mackay, A. Macleod, T. Maher, M. Marquarding, N. McClure-Griffiths, D. McConnell, S. Mickle, P. Mirtschin, R. Norris, S. Neuhold, A. Ng, J. O'Sullivan, J. Pathikulangara, S. Pearce, C. Phillips, R. Y. Qiao, J. E. Reynolds, A. Rispler, P. Roberts, D. Roxby, A. Schinckel, R. Shaw, M. Shields, M. Storey, T. Sweetnam, E. Troup, B. Turner, A. Tzioumis, T. Westmeier, M. Whiting, C. Wilson, T. Wilson, K. Wormnes, and X. Wu, "The australian square kilometre array pathfinder: System architecture and specifications of the boolardy engineering test array," *Publications of the Astronomical Society of Australia*, vol. 31, 2014.

[9] D. McConnell, J. R. Allison, K. Bannister, M. E. Bell, H. E. Bignall, A. P. Chippendale, P. G. Edwards, L. Harvey-Smith, S. Hegarty, I. Heywood, A. W. Hotan, B. T. Indermuehle, E. Lenc, J. Marvil, A. Popping, W. Raja, J. E. Reynolds, R. J. Sault, P. Serra, M. A. Voronkov, M. Whiting, S. W. Amy, P. Axtens, L. Ball, T. J. Bateman, D. C.-J. Bock, R. Bolton, D. Brodrick, M. Brothers, A. J. Brown, J. D. Bunton, W. Cheng, T. Cornwell, D. DeBoer, I. Feain, R. Gough, N. Gupta, J. C. Guzman, G. A. Hampson, S. Hay, D. B. Hayman, S. Hoyle, B. Humphreys, C. Jacka, C. A. Jackson, S. Jackson, K. Jeganathan, J. Joseph, B. S. Koribalski, M. Leach, E. S. Lensson, A. MacLeod, S. Mackay, M. Marquarding, N. M. McClure-Griffiths, P. Mirtschin, D. Mitchell, S. Neuhold, A. Ng, R. Norris, S. Pearce, R. Y. Qiao, A. E. T. Schinckel, M. Shields, T. W. Shimwell, M. Storey, E. Troup, B. Turner, J. Tuthill, A. Tzioumis, R. M. Wark, T. Westmeier, C. Wilson, and T. Wilson, "The australian square kilometre array pathfinder: Performance of the boolardy engineering test array," *Publications of the Astronomical Society of Australia*, vol. 33, 2016.

[10] M. Elmer, B. D. Jeffs, K. F. Warnick, J. R. Fisher, and R. D. Norrod, "Beamformer design methods for radio astronomical phased array feeds," *IEEE Transactions on Antennas and Propagation*, vol. 60, no. 2, pp. 903–914, Feb 2012.

[11] C. Clapham and J. Nicholson, *The Concise Oxford Dictionary of Mathematics (Oxford Quick Reference)*. OUP Oxford, 2009. [Online]. Available: https://www.amazon.com/Concise-Oxford-Dictionary-Mathematics-Reference-ebook/dp/B004OEK3BE?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=B004OEK3BE

[12] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing, Second Edition*. Cambridge University Press, 1992. [Online]. Available: https://www.amazon.com/Numerical-Recipes-Scientific-Computing-Second/

dp/0521431085?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0521431085

[13] C. F. G. Gene H., Van Loan, *Matrix Computations. Third Edition.* The Johns Hopkins University Press, 1996. [Online]. Available: https://www.amazon.com/Matrix-Computations-Third-Charles-Golub/dp/B007CV78PU?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=B007CV78PU

[14] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *Ann. Math. Statist.*, vol. 21, no. 1, pp. 124–127, 03 1950. [Online]. Available: https://doi.org/10.1214/aoms/1177729893

[15] *Applied Parallel and Scientific Computing.* Springer Berlin Heidelberg, 2013. [Online]. Available: http://www.ebook.de/de/product/20237877/applied_parallel_and_scientific_computing.html

[16] I. LaRoche and S. Roy, "An efficient regular matrix inversion circuit architecture for mimo processing," in *2006 IEEE International Symposium on Circuits and Systems*, May 2006, pp. 4 pp.–.

[17] M. Ylinen, A. Burian, and J. Takala, "Updating matrix inverse in fixed-point representation: direct versus iterative methods," in *Proceedings. 2003 International Symposium on System-on-Chip (IEEE Cat. No.03EX748)*, Nov 2003, pp. 45–48.

[18] Y. B. Mahdy, S. A. Ali, and K. M. Shaaban, "Algorithm and two efficient implementations for complex multiplier," in *Electronics, Circuits and Systems, 1999. Proceedings of ICECS '99. The 6th IEEE International Conference on*, vol. 2, Sep 1999, pp. 949–952 vol.2.

[19] N. J. Higham, *Accuracy and Stability of Numerical Algorithms.* Soc for Industrial & Applied Math, 1996. [Online]. Available: https://www.amazon.com/Accuracy-Stability-Numerical-Algorithms-Nicholas/dp/0898713552?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0898713552

[20] J. Ding and G. Yao, "The eigenvalue problem of a specially updated matrix," vol. 185, pp. 415–420, 02 2007.