Phased-Array Feed Instrumentation and Processing for Astronomical Detection, Interference Mitigation, and Transient Parameter Estimation

Richard Allen Black
Brigham Young University

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Electrical and Computer Engineering Commons

BYU ScholarsArchive Citation
https://scholarsarchive.byu.edu/etd/6627

This Dissertation is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
Phased Array Feed Instrumentation and Processing for Astronomical Detection,
Interference Mitigation, and Transient Parameter Estimation

Richard Allen Black

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Brian D. Jeffs, Chair
Karl F. Warnick
Neal K. Bangerter
Michael Rice
David G. Long

Department of Electrical and Computer Engineering
Brigham Young University

Copyright © 2017 Richard Allen Black
All Rights Reserved
ABSTRACT

Phased Array Feed Instrumentation and Processing for Astronomical Detection, Interference Mitigation, and Transient Parameter Estimation

Richard Allen Black
Department of Electrical and Computer Engineering, BYU
Doctor of Philosophy

Radio astronomy, the survey and study of naturally occurring astronomical radio signals, is a challenging field in terms of engineering requirements. The typical astronomical signal of interest is incredibly faint, resulting in very low signal-to-noise ratios (SNRs) on the order of -30 dB or lower. To detect such signals, one must have an uncommonly low and stable receiver noise temperature, high gain through large aperture reflectors, and state-of-the-art signal processing algorithms. One must also be able to mitigate the effects of interference, the presence of which, even if extremely weak, can completely mask the faint astronomical signals of interest. To this end, this work presents the development of and results from a new broadband phased array feed (PAF) named the Focal L-Band Array for the Green Bank Telescope (FLAG). This instrument is able to form multiple simultaneous beams to survey a large patch of sky instantaneously, and has a minimum system noise temperature ($T_{sys}$) of 16.83 K. This PAF also has the potential to use spatial filtering techniques to place pattern nulls in the direction of interfering signals through the use of an orthogonal projection. This work will also present an improved method for computing an orthogonal projection operator, which is able to place a spatially broad null in the direction of a moving RFI source. A formal derivation of some detection and estimation theory properties for astronomical radio transients is also presented, which formalization is lacking within the astronomical community. This includes maximum-likelihood detectors and estimators and a Cramér Rao bound (CRB) analysis of astronomical transient parameters.

Keywords: array systems, radio astronomy, radio frequency interference, pulsars, fast radio bursts, phased array feed
ACKNOWLEDGMENTS

A work of this magnitude does not come about without the contributions and support of several people. In this case, this dissertation represents the culmination of nearly 30 years of concerted effort by loving parents, dedicated teachers, encouraging colleagues, and a supportive spouse. Without their sacrifices on my behalf, this work would not exist.

I would first like to acknowledge my graduate advisor Dr. Brian Jeffs, who challenged me to stretch beyond my perceived limits and taught me how to critically analyze my own work. He was a consistent champion for my ideas and research, even when receiving disappointing news about multiple rejected manuscripts. His interest in my life beyond graduate school helped me feel like a fellow colleague, which gave me the confidence to propose new and crazy ideas without fear of rebuke. The same holds true for the other faculty members on my committee, namely Karl Warnick, Neal Bangerter, Michael Rice, and David Long.

My parents Guy and Maria deserve more recognition than they currently receive for their many sacrifices in making me who I am today. Once in elementary school, my teacher informed them that I “would never pull [my] weight in math.” In response, my parents challenged me every night, drilling me until the math became second-nature. Without their diligence, I would not have pursued engineering as my career path or even considered a doctorate degree. Furthermore, their frequent willingness to babysit my children gave me with precious time to write.

Lastly, words cannot begin to describe how much my wonderful wife Karen contributed to the completion of this work. For the last five years, she has supported me emotionally, intellectually, and financially simply because she loved me and wanted to support me. During this time, she gave birth to our two children Chloe and Sam and stayed home with them while remotely working a full-time job. At times, I felt like I was putting an undue burden on her, a feeling which she quickly dispelled without complaint. My life has been infinitely better since she came into my life. I love her, and I look forward to the day in which I can, in some small way, repay her for all that she has done for me.
# Table of Contents

## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Chapter 1 Introduction

1.1 Problem Statements ........................................... 3

1.2 Related Work .................................................. 4

   1.2.1 Phased Array Feeds ...................................... 4
   1.2.2 Transient Detection and Estimation ...................... 5
   1.2.3 Interference Mitigation ................................. 6

1.3 Dissertation Contributions ................................. 7

   1.3.1 Focal L-Band Array for the Green Bank Telescope Development .............................. 7
   1.3.2 Astronomical Transient Detection and Estimation Theory ................................. 8
   1.3.3 Radio Frequency Mitigation Algorithms ...................... 9

1.4 Dissertation Outline ......................................... 9

## Chapter 2 Background

2.1 Introduction .................................................. 11

2.2 Phased Array Feed Signal Model ............................. 11

2.3 Beamformer Weights .......................................... 14

   2.3.1 Maximum Gain Beamformer ............................... 15
   2.3.2 Maximum SNR Beamformer ................................. 16
   2.3.3 Linear Constraint Minimum Variance Beamformer ............ 16

2.4 Array Feed Calibration ...................................... 17

2.5 Fundamentals of Astronomical Transients .................. 19

2.6 Detection of Astronomical Transients ..................... 21

2.7 Estimation of Astronomical Transient Parameters ........ 22

2.8 Radio-Frequency Interference Mitigation Methods ........ 22

   2.8.1 Orthogonal Projections ................................. 23

## Chapter 3 Maximum Likelihood Detection and Estimation for Phased Array Feeds

3.1 Introduction .................................................. 25

3.2 Maximum Likelihood Framework ............................... 26

   3.2.1 Useful Theorems ....................................... 27

3.3 Array Steering Vector Estimator ............................ 28

3.4 Transient Detection in Element Space .......................... 30

   3.4.1 Spatially White Noise ................................. 32
   3.4.2 Calibrated Array .................................... 33
   3.4.3 Uncalibrated Array .................................. 33

3.5 Transient Detection in Beam Space .......................... 34

   3.5.1 Spatially White Noise ................................. 36
   3.5.2 Calibrated Array .................................... 37
### Chapter 4 Cramér Rao Bound for Astronomical Transient Parameters

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>39</td>
</tr>
<tr>
<td>4.2</td>
<td>Signal Model</td>
<td>41</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Single-Pixel Model</td>
<td>42</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Array Antenna Model</td>
<td>43</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Pulse Shape Model</td>
<td>44</td>
</tr>
<tr>
<td>4.3</td>
<td>Framework for Cramér Rao Bound</td>
<td>44</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Single-Pixel Antenna</td>
<td>45</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Array Antenna</td>
<td>46</td>
</tr>
<tr>
<td>4.4</td>
<td>Methods</td>
<td>48</td>
</tr>
<tr>
<td>4.5</td>
<td>Results</td>
<td>48</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Window Size</td>
<td>49</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Pulse Start Time</td>
<td>50</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Dispersion Measure</td>
<td>51</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Frequency</td>
<td>52</td>
</tr>
<tr>
<td>4.5.5</td>
<td>Contiguous Bandwidth</td>
<td>53</td>
</tr>
<tr>
<td>4.5.6</td>
<td>Fractional Non-Contiguous Bandwidth</td>
<td>53</td>
</tr>
<tr>
<td>4.5.7</td>
<td>Frequency Resolution</td>
<td>56</td>
</tr>
<tr>
<td>4.5.8</td>
<td>Pulse Width</td>
<td>56</td>
</tr>
<tr>
<td>4.5.9</td>
<td>Temporal Resolution</td>
<td>57</td>
</tr>
<tr>
<td>4.5.10</td>
<td>Signal-to-Noise Ratio</td>
<td>59</td>
</tr>
<tr>
<td>4.5.11</td>
<td>Angle of Arrival</td>
<td>60</td>
</tr>
<tr>
<td>4.6</td>
<td>Conclusions</td>
<td>62</td>
</tr>
</tbody>
</table>

### Chapter 5 Focal L-Band Array for the Green Bank Telescope Back End

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>66</td>
</tr>
<tr>
<td>5.2</td>
<td>Operational Modes</td>
<td>68</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Coarse-Channel Mode</td>
<td>69</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Fine-Channel Mode</td>
<td>70</td>
</tr>
<tr>
<td>5.3</td>
<td>HPC Specifications</td>
<td>70</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Graphics Cards</td>
<td>71</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Network Cards</td>
<td>72</td>
</tr>
<tr>
<td>5.4</td>
<td>HASHPIPE</td>
<td>72</td>
</tr>
<tr>
<td>5.5</td>
<td>GPU Libraries</td>
<td>73</td>
</tr>
<tr>
<td>5.5.1</td>
<td>xGPU</td>
<td>74</td>
</tr>
<tr>
<td>5.6</td>
<td>July 2016 Commissioning Results</td>
<td>74</td>
</tr>
<tr>
<td>5.7</td>
<td>May 2017 Commissioning Results</td>
<td>81</td>
</tr>
<tr>
<td>5.7.1</td>
<td>Outdoor Test Facility Experiments</td>
<td>81</td>
</tr>
<tr>
<td>5.7.2</td>
<td>On-Telescope</td>
<td>82</td>
</tr>
<tr>
<td>5.8</td>
<td>August 2017 Commissioning Results</td>
<td>89</td>
</tr>
<tr>
<td>5.9</td>
<td>Conclusions and Future Work</td>
<td>100</td>
</tr>
</tbody>
</table>
Chapter 6  Deep Broad Null Formation via Orthogonal Projection and Subspace Tracking  
6.1 Introduction ................................................................. 103
6.2 Signal Model and Algorithm Development .......................... 106
6.3 Projection-Based RFI Mitigation ...................................... 109
   6.3.1 Background .......................................................... 109
   6.3.2 A Framework for Existing RFI Projection Algorithms ....... 110
   6.3.3 Proposed Algorithm for Broad Null Formation ................. 114
6.4 Simulation ................................................................. 118
   6.4.1 Simulation Details .................................................. 118
   6.4.2 Performance Metrics .............................................. 120
   6.4.3 Results .............................................................. 122
   6.4.4 Canceling Multiple RFI .............................................. 127
   6.4.5 STI Length Considerations ...................................... 127
   6.4.6 LTI Length Considerations ...................................... 131
6.5 Conclusion ............................................................... 131

Chapter 7  Conclusions and Future Work ................................. 133

References ........................................................................... 135

Appendix A  Derivations of Jacobians ................................... 146
   A.1 Jacobian for Single Antenna Model ................................. 146
   A.2 Jacobian for Array Antenna Model ................................. 147

Appendix B  FLAG Development Details ................................. 151
   B.1 xGPU Details ........................................................... 151
   B.2 HASHPIPE Plugins ...................................................... 151
   B.3 HASHPIPE Threads ...................................................... 152
      B.3.1 flag_net_thread ...................................................... 155
      B.3.2 flag_transpose_thread .............................................. 156
      B.3.3 flag_pftranspose_thread .......................................... 157
      B.3.4 flag_frb_transpose_thread ....................................... 157
      B.3.5 flag_bx_transpose_thread ........................................ 158
      B.3.6 flag_pf_thread ....................................................... 158
      B.3.7 flag_correlator_thread ............................................ 158
      B.3.8 flag_frb_correlator_thread ....................................... 159
      B.3.9 flag_pf_correlator_thread ......................................... 159
      B.3.10 flag_beamform_thread ............................................ 159
      B.3.11 flag_bx_thread ...................................................... 160
      B.3.12 flag_bfx_thread .................................................... 160
      B.3.13 flag_corrsave_thread ............................................. 160
      B.3.14 flag_frb_corrsave_thread ......................................... 161
      B.3.15 flag_pfbsave_thread .............................................. 161
## LIST OF TABLES

4.1 Major Effects of Parameters on CRB for DM ............................................. 64
5.1 HPC Specifications ...................................................................................... 71
5.2 PCI-E Slot Population (out-of-date) .............................................................. 71
5.3 GeForce GTX 780 Ti Specifications .............................................................. 72
5.4 GeForce GTX 980 Ti Specifications .............................................................. 72
6.1 Performance on GBT without Auxiliary ....................................................... 126
6.2 Performance on GBT with Auxiliary ............................................................ 127
6.3 Data Rates for Various Approaches ............................................................... 130
B.1 Real-Time Beamformer Metadata .............................................................. 160
B.2 FLAG Back End Operational Mode Table ................................................... 164
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Artistic rendition of a pulsar, depicting magnetic field lines and radiation pattern.</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>PAFs are calibrated by steering the dish across a strong point source and estimating the corresponding steering vector vectors for each pointing. The red dots indicate an “on” pointing, and the blue dots indicate an “off” pointing.</td>
<td>18</td>
</tr>
<tr>
<td>4.1</td>
<td>Effect of total window size on CRB. The vertical line indicates when the pulse was contained in the window (i.e., $t_0 + T$).</td>
<td>50</td>
</tr>
<tr>
<td>4.2</td>
<td>Effect of pulse start time $t_0$ on CRB. The region within the vertical lines is where the processing window contains at least 90% of the pulse power.</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>Effect of Dispersion Measure on CRB.</td>
<td>52</td>
</tr>
<tr>
<td>4.4</td>
<td>Effect of observation frequency on CRB.</td>
<td>53</td>
</tr>
<tr>
<td>4.5</td>
<td>Effect of total bandwidth on CRB.</td>
<td>54</td>
</tr>
<tr>
<td>4.6</td>
<td>Tested frequency channel subsets.</td>
<td>55</td>
</tr>
<tr>
<td>4.7</td>
<td>CRB for the tested frequency channel subsets.</td>
<td>56</td>
</tr>
<tr>
<td>4.8</td>
<td>Effect of observation frequency resolution on CRB.</td>
<td>57</td>
</tr>
<tr>
<td>4.9</td>
<td>Effect of pulse width on CRB.</td>
<td>58</td>
</tr>
<tr>
<td>4.10</td>
<td>Effect of temporal resolution on CRB.</td>
<td>59</td>
</tr>
<tr>
<td>4.11</td>
<td>Effect of pulse width on CRB for various temporal resolutions.</td>
<td>60</td>
</tr>
<tr>
<td>4.12</td>
<td>The temporal resolution that minimizes the CRB for a given pulse width.</td>
<td>61</td>
</tr>
<tr>
<td>4.13</td>
<td>Effect of arrival angle on CRB with lower S/N.</td>
<td>62</td>
</tr>
<tr>
<td>4.14</td>
<td>Effect of arrival angle on CRB.</td>
<td>63</td>
</tr>
<tr>
<td>4.15</td>
<td>Effect of arrival angle on CRB with lower S/N.</td>
<td>64</td>
</tr>
<tr>
<td>5.1</td>
<td>High-level analog receiver and digital processor block diagram.</td>
<td>67</td>
</tr>
<tr>
<td>5.2</td>
<td>A single GBT-2 dipole.</td>
<td>67</td>
</tr>
<tr>
<td>5.3</td>
<td>Block diagram describing the various operational modes.</td>
<td>69</td>
</tr>
<tr>
<td>5.4</td>
<td>A photo of one of the five HPCs.</td>
<td>70</td>
</tr>
<tr>
<td>5.5</td>
<td>A generic HASHPIPE program block diagram.</td>
<td>73</td>
</tr>
<tr>
<td>5.6</td>
<td>The correlator output for a single frequency channel.</td>
<td>74</td>
</tr>
<tr>
<td>5.7</td>
<td>A picture of the array feed in the outdoor test facility prior to installation on the dish.</td>
<td>75</td>
</tr>
<tr>
<td>5.8</td>
<td>The sensitivity map for a calibration grid on 3C295.</td>
<td>77</td>
</tr>
<tr>
<td>5.9</td>
<td>The array element layout as seen from the rear.</td>
<td>78</td>
</tr>
<tr>
<td>5.10</td>
<td>Element pattern maps from a calibration grid on 3C295.</td>
<td>79</td>
</tr>
<tr>
<td>5.11</td>
<td>The formed beam layout on the sky.</td>
<td>80</td>
</tr>
<tr>
<td>5.12</td>
<td>Formed beam pattern maps from a calibration grid on 3C295.</td>
<td>80</td>
</tr>
<tr>
<td>5.13</td>
<td>Measured $T_{sys}$ with bit underflow.</td>
<td>83</td>
</tr>
<tr>
<td>5.14</td>
<td>Measured $T_{sys}$ with improved re-quantization.</td>
<td>83</td>
</tr>
<tr>
<td>5.15</td>
<td>The X-polarization sensitivity map for a calibration grid on 3C295.</td>
<td>85</td>
</tr>
<tr>
<td>5.16</td>
<td>The Y-polarization sensitivity map for a calibration grid on 3C295.</td>
<td>85</td>
</tr>
<tr>
<td>5.17</td>
<td>X-polarization $T_{sys}/\eta$ spectrum on 3C295.</td>
<td>86</td>
</tr>
<tr>
<td>5.18</td>
<td>Y-polarization $T_{sys}/\eta$ spectrum on 3C295.</td>
<td>86</td>
</tr>
<tr>
<td>5.19</td>
<td>X-polarization element pattern maps from a calibration grid on 3C295.</td>
<td>87</td>
</tr>
</tbody>
</table>
5.20 Y-polarization element pattern maps from a calibration grid on 3C295. ................. 88
5.21 X-polarization formed beam maps from a calibration grid on 3C295. ..................... 89
5.22 Y-polarization formed beam maps from a calibration grid on 3C295. ..................... 90
5.23 X-polarization sensitivity map for a calibration grid on 3C295. ......................... 91
5.24 Y-polarization sensitivity map for a calibration grid on 3C295. ......................... 91
5.25 X-polarization $T_{sys}/\eta$ spectrum on 3C295. ............................................ 92
5.26 Y-polarization $T_{sys}/\eta$ spectrum on 3C295. ............................................ 92
5.27 X-polarization element pattern maps from a calibration grid on 3C295. ................ 94
5.28 Y-polarization element pattern maps from a calibration grid on 3C295. ................ 95
5.29 X-polarization formed beam maps from a calibration grid on 3C295. .................... 96
5.30 Y-polarization formed beam maps from a calibration grid on 3C295. .................... 96
5.31 Map of available constraint points for each beam. ........................................... 97
5.32 Linearly constrained beam patterns using six degrees of freedom. ....................... 98
5.33 Linearly constrained beam patterns using ten degrees of freedom. ....................... 99
5.34 Linearly constrained beam patterns using 15 degrees of freedom. ....................... 100
5.35 X-polarization $T_{sys}/\eta$ broad spectrum on 3C123. .................................... 101
5.36 Y-polarization $T_{sys}/\eta$ broad spectrum on 3C123. .................................... 101

6.1 An example of subspace smearing. ............................................................... 104
6.2 A motion-agnostic approach to RFI mitigation that is ineffective when the RFI is moving quickly. Depending on the application, the desired data product is either the RFI-cleaned correlation $\hat{R}_x$ (e.g., for synthesis imaging) or projection operator (e.g., for beamformer weight calculation and updates). ................................. 110
6.3 A motion-agnostic approach when augmented with an auxiliary antenna that tracks the RFI. (1) indicates that only the matrix entries corresponding to the array correlations (i.e., $\hat{R}_x$ from (6.13)) are propagated on this line, and (2) indicates that only the correlations between the array and the auxiliary antenna (i.e., $\hat{r}_{xa}$ from (6.13)) are propagated on this line. ................................................. 111
6.4 Traditional solution to moving RFI mitigation that trades off the integration length of the sample correlation matrix to mitigate motion at the expense of increased sample estimation error. (1) indicates that only the matrix entries corresponding to the array correlations (i.e., $\hat{R}_x$ from (6.13)) are propagated on this line, and (2) indicates that only the correlations between the array and the auxiliary antenna (i.e., $\hat{r}_{xa}$ from (6.13)) are propagated on this line. ................................................. 112
6.5 The proposed broad-null-forming solution to moving RFI mitigation that is less sensitive to motion smearing of the RFI subspace, and has low sample estimation error. . 114
6.6 The proposed broad-null-forming solution to moving RFI mitigation that, through the use of an auxiliary antenna, requires less computational resources, is less sensitive to RFI motion, and has low sample estimation error. ................................. 114
6.7 The FLAG 19-element dual-polarization dipole array. ..................................... 120
6.8 The interference power level regions in which the performance of an RFI projection is primarily limited by a particular effect. ................................................. 121
6.9 The performance of the three approaches on the GBT with $Q = 1$ for the motion-agnostic and motion-cognizant approaches and $Q = 2$ for the broad null approach, an LTI of one second, an STI of 0.01 seconds, and GPS satellite BIIR-11 (PRN 19). . . 123
6.10 The performance of the three approaches on the GBT with $Q = 2$ for each approach, an LTI of one second, an STI of 0.01 seconds, and GPS satellite BIIR-11 (PRN 19). 124

6.11 The performance of the three approaches on the assisted GBT with $Q = 1$ for the motion-agnostic and motion-cognizant approaches and $Q = 2$ for the broad null approach, an LTI of one second, an STI of 0.01 seconds, and GPS satellite BIIR-11 (PRN 19). 125

6.12 The effect of the algorithms on the SNR, with $Q = 1$ for the motion-agnostic and motion-cognizant approaches and $Q = 2$ for the broad null. 126

6.13 The performance of the three approaches on the unassisted GBT with $Q = 4$ for all algorithms, an LTI of one second, an STI of 0.01 seconds, and two RFI sources: GPS satellites BIIR-11 (PRN 19) and BIIR-13 (PRN 02). 128

6.14 The performance of the three approaches on the assisted GBT with $Q = 2$ for the motion-cognizant and agnostic algorithms and $Q = 4$ for the broad null, an LTI of one second, an STI of 0.01 seconds, and two RFI sources: GPS satellites BIIR-11 (PRN 19) and BIIR-13 (PRN 02). 129

6.15 The performance of the broad null and motion-agnostic approach with $Q = 2$ and varying STI lengths. The LTI length is one second, and the interfering source is GPS satellite BIIR-11 (PRN 19). 130

6.16 The performance of the broad null approach when the LTI length is varied. 132

B.1 The HASHPIPE thread layout for the FLAG back end. 153

B.2 The state machine diagram for flag_net_thread. 156

B.3 The data coming out of the flag_net_thread are ordered such each block contains 200 frames, which each contain eight ROACH packets. Each packet contains 20 time samples of 25 frequency channels across eight antenna elements. 156

B.4 The state machine diagram for every thread except flag_net_thread. 157

B.5 The data format coming out of the flag_transpose_thread. Each block contains 4000 time samples of 25 frequency channels across 64 antenna elements. 157

B.6 The data format coming out of the flag_correlator_thread. 159

B.7 Block diagram depicting the control interfaces between the Dealer, Player, HASHPIPE codes, FITS writer, and shared memory. 163

B.8 The file format for the beamformer weights. 171
CHAPTER 1. INTRODUCTION

Radio astronomy is the scientific survey and study of naturally occurring astronomical radio signals that are emitted from various celestial objects such as stars, galaxies, and optically invisible bodies such as super nova remnants, quasars, pulsars, and gasses like neutral hydrogen clouds. There are many ways that natural radio waves are induced. For example, when a photon collides with a neutral hydrogen atom, it can cause the atom’s electron to be ejected from its orbit. A new electron then drops from a higher energy orbit and emits a photon with a frequency of 1420.406 MHz that can then be detected on Earth. Deep-space signals like this are incredibly weak with signal-to-noise ratios (SNRs) often lower than -30 dB or more and require high-gain, low-noise receiver systems to have any hope of being detected. To make matters worse, these signals are becoming harder to detect as the number of man-made sources of radio frequency interference (RFI), such as orbiting satellites and cell phone towers, continues to increase and masks astronomical signals.

One unique class of astronomical signal is the radio transient, a short broadband burst that undergoes a frequency-dependent time delay as it propagates through the diffuse turbulent plasma between star systems known as the interstellar medium (ISM) [1]. This phenomenon, known as dispersion, causes a deep-space pulse to have a frequency-time structure similar to that of an frequency-modulated (FM) radar chirp where lower frequency components experience a greater delay while propagating through the ISM. Figure 1.1a depicts an example of such a pulse. This structure is fully characterized by a single parameter called the dispersion measure (DM), which is defined as the number of free electrons in a square-centimeter column along the line of sight between the source and observer, and is measured in cm$^{-3}$ pc, where “pc” stands for parsec [1].

A sub-class of these transients is the pulsar, a highly magnetized, rapidly rotating neutron star that emits a narrow beam (columbated by the poles of the magnetic field) of broadband energy. This behavior causes pulsars to behave much like radio lighthouses, as depicted in Figure 1.1.
Consequently, received pulsar emissions are steadily periodic, making them powerful probes of physical principles ranging from gravitational field and wave theory to solid state physics [1].

Related to pulsars, the fast radio burst (FRB) has become of significant interest as of late in the astronomical community. Like pulsars, FRBs are broadband pulses which undergo the dispersion phenomenon, but they are non-repeating, exhibit relatively high energy, and have millisecond-scale pulse widths. Astronomers have not yet concluded what causes these one-time dispersed bursts, but they estimate that more than 2500 FRBs occur in the observable sky each day [2], [3]. Despite this hypothesized high rate of FRB events, only 23 have been observed to date [4]–[17].
In this work, we seek to increase the likelihood of FRB detection in radio astronomical observations by developing new array sensor technology, which, through the formation of multiple simultaneous beams, will greatly expand the instantaneous field of view (FoV) of radio telescopes. This work will also expand array sensor signal processing and interference cancelation theory and find new ways to detect and characterize astronomical transients.

1.1 Problem Statements

There are three primary problems that this dissertation aims to address: (1) the hardware and software challenges inherent in a high-bandwidth, numerous-element phased array feed signal processing system, (2) the lack of formal detection and estimation theory for radio astronomical transients such as pulsars and FRBs, and (3) the obstacles inherent in nulling rapidly moving, man-made RFI with an array system.

Problem Statement: Phased Array Feeds

The dearth of confirmed FRB detections is likely due to the prominent use of single-element feeds on radio telescope dishes, which are only able to survey a single sky pointing at a given time. These feeds are sometimes called “single-pixel” feeds, and they have very narrow FoVs. One can greatly widen the FoV but retain the high gain of a large dish aperture by using phased array feeds (PAFs), consisting of tightly packed antenna elements placed at the focal point of a parabolic reflector. PAF systems are able to form multiple simultaneous beams on the sky, thus creating a “radio camera” of sorts [18]. The wider FoV that PAFs provide, however, comes at a significant hardware and software development cost since PAF systems are incredibly complex systems.

Problem Statement: Transient Detection and Estimation

Most transient detection algorithms in the literature require a brute force search over trial DMs and pulse widths, resulting in a significant computational burden [19], [20]. For example, the current practice for detecting an astronomical transient is to “de-disperse,” or introduce frequency-dependent delays and sum across, the pulse exhaustively across a large number of trial DM values and to search for peaks [1], [21], [22].
Much of the radio astronomy community has not viewed detection of astronomical transients and estimation of their associated parameters from a statistical signal processing perspective. With a formal detection and estimation theoretic treatment, it will be possible to compare current algorithm performance to theoretical limits and perhaps suggest algorithm improvements in terms of accuracy and computational efficiency.

**Problem Statement: Radio Frequency Interference**

Transients and other weak astronomical signals are also highly susceptible to RFI from sources such as ground-based transmitter stations and orbiting Global Positioning System (GPS) and communication satellites [23]. Traditionally, astronomers simply handle RFI by either identifying (sometimes called “flagging”) and discarding the affected data samples or avoiding it by scanning in regions with little RFI. However, there is a growing consensus that this will not be a viable strategy moving forward as some astronomers are reporting 100% data loss in some scientifically important frequencies [24]–[26]. PAF technology also offers a unique solution to RFI cancelation through the spatial filtering capabilities of beamforming.

Despite this growing RFI problem and the already available technology to address it, astronomers are wary of implementing spatial filtering techniques for fear of artificially modifying the true underlying signal of interest. It may prove that this fear will only be alleviated by developing and implementing new and existing RFI mitigation algorithms in PAF digital receiver systems and demonstrating proper operation. Furthermore, the continual development of novel, cost-effective RFI mitigation algorithms can help make adoption easier.

1.2 Related Work

1.2.1 Phased Array Feeds

There have been some groups that have successfully developed and deployed PAF systems. For example, the Dominion Radio Astrophysical Observatory (DRAO) developed a demonstration PAF named the Phased-Array Feed Demonstrator (PHAD) [27], [28]. The system consists of 180 Vivaldi antenna elements and is able to process up to 32 MHz of instantaneous bandwidth.
The research group at BYU also previously developed two PAF receivers, a narrowband demonstrator system [29]–[31] and a midband one known as the x64 system [32], [33]. The narrowband system supports 40 antenna elements across a 450-kHz bandwidth and can acquire data continuously for 60 seconds. This system was deployed for L-band observations (1-2 GHz) on the 20-meter-diameter dish at the National Radio Astronomy Observatory (NRAO) [34], [35] and on the 300-meter-diameter in-ground dish at the Arecibo Observatory [36]. The x64 system supported 64 antennas across a bandwidth of 6 MHz for an indefinite period of time. This system was deployed at the Arecibo Observatory for L-band observations [37], [38] and on the 100-meter-diameter, fully steerable Green Bank Telescope for W-band (75-110 GHz) [39].

While the above systems represent significant strides in the development of PAFs, they are primarily proof-of-concept systems and engineering prototypes. In order to be viable for major astronomical scientific use, these PAF receivers must be able to support significantly more bandwidth, be supported by large data storage libraries, permanent reliable hardware, and user-friendly software human interfaces.

There are other in-development PAFs that are approaching scientific viability in terms of sensitivity and bandwidth support. One such instrument is the Aperture Tile In Focus (APERTIF) array for the Westerbrook Synthesis Radio Telescope (WSRT) in the Netherlands [40]–[42]. The APERTIF instrument will support 121 Vivaldi antenna elements across a 300-MHz analog bandwidth. The Australian Square Kilometre Array Pathfinder (ASKAP) MkII PAF [43]–[46] in Australia is a now fully operational 300-MHz 36-dish large interferometric imaging array, with each dish equipped with a 188-element array.

1.2.2 Transient Detection and Estimation

The current gold standard for transient parameter estimation is to use the pulsar-timing package TEMPO2, a sophisticated model-fitting software package created by Hobbes et al. that estimates pulsar right ascension (RA) angle, declination (DEC) angle, pulse frequency and its derivatives, and DM [47], [48]. Hobbes et al. also showed that several existing timing models had timing errors, which effectively results in non-white measurement noise [49]. Using the timing model in TEMPO2, many others have studied the effects of non-ideal parameters, such as time-varying DMs studied by You et al. [50]. Coles et al. and Haasteren et al. studied low-frequency
temporal correlations, or “red timing noise” in timing residuals and updated techniques to perform pulsar timing in the presence of said such correlated noise [51], [52]. Cordes et al. examined the effects of frequency-dependent DMs, or “chromatic DMs,” on pulsar timing [53]. Haasteren also developed an accelerated approach to pulsar timing using compressed sensing techniques [52].

All of these works have advanced the capability of pulsar timing instruments, but they primarily focus on improving the pulsar timing model and attempt to improve model fitting algorithms instead of using formal estimation theory approaches and analysis tools, such as the Cramér Rao bound (CRB). Furthermore, these approaches have all been for single-beam instruments and have not yet explored the increased value of a PAF.

1.2.3 Interference Mitigation

A wealth of previous work exists for spatial filtering RFI mitigation techniques using array systems. Frost has shown that one can use linear constraint minimum variance (LCMV) beamformers to create patterns with nulls in prescribed directions if the corresponding array constraint vectors are known [54]. Leshem and van der Veen introduced the idea of using orthogonal projection operators to project received signal vectors onto the orthogonal complement of the RFI subspace, which is estimated via an eigenvector decomposition [55], [56]. This approach is known as “subspace projection.” Jeffs et al. enhanced the subspace projection algorithm by incorporating an RFI-tracking auxiliary antenna. This is done by cross-correlating the auxiliary signal with those of the array sensor, providing a high interference-to-noise ratio (INR) estimate of the RFI subspace making low-power RFI subspace projection more robust [57], [58]. The performance of auxiliary-assisted subspace projection could also be enhanced with an adaptive beamforming array as demonstrated by Sardarabadi et al. [59]. These approaches were then formalized for other types of array systems such as PAFs by Jeffs et al. [18], [60].

Hellbourg studied the effects of RFI motion, also known as subspace smearing where the RFI is not well characterized by a single basis vector, on the effectiveness of subspace projection [61]. Landon et al. created a computationally expensive reduced-parameter subspace tracking algorithm that effectively reconstructs the array steering vector at any given time sample [62].
al. then showed that the effects of large spacings and bandwidths can be somewhat minimized by using subspace projection on sub-arrays instead of the entire array [63].

These projection operators can induce undesirable signal vector scaling and rotation, which can be corrected using an on-average subspace bias correction matrix introduced by Raza et al. [56]. Bias introduced into an RFI-mitigated power spectrum by subspace projection in a beam-forming array was addressed by Jeffs et al., who created an revised power spectrum estimator that incorporates the subspace bias correction matrix [18], [64], [65].

All of these contributions, while demonstrably effective in simulation, lack significant experimental results involving real radio astronomical observations. This is likely due to their strict requirements of extremely narrow bandwidths and short correlator dump times, which makes adoption of these algorithms incredibly expensive. Consequently, the astronomy community has been reluctant to adopt these RFI-mitigation algorithms out of fear of artificially modifying the weak signals of interest. This fear may continue until there are several concrete demonstrations of these techniques.

1.3 Dissertation Contributions

In this dissertation, the author’s contributions, in no particular order, are summarized as follows:

1.3.1 Focal L-Band Array for the Green Bank Telescope Development

• Led student development team to develop 150-MHz bandwidth GPU-based beamformer and frequency channelizer.

• Created a GPU-based code to compute time-average power across the entire 150-MHz bandwidth.

• Integrated GPU-based codes and existing correlator library named “xGPU” into thread-based pipeline software named “HASHPIPE.”

• Integrated HASHPIPE codes into existing Green Bank control codes named the “Digital Backend (DIBAS).
• Created packet format definition for FPGA-to-server-PC data transfer.

• Configured large 40-gigabit-Ethernet (40 GbE) network for fast packet forwarding.

• Deployed reduced-bandwidth version of the FLAG system for initial PAF commissioning (August 2016).

• Developed MATLAB-based data processing codes.

• Integrated the full system.

• Improved telescope-to-back-end communication codes for real-time control.

• Deployed full-bandwidth version of the FLAG system for second PAF commissioning (May 2017).

• Corrected system bugs in preparation for final PAF commissioning (July 2017).

1.3.2 Astronomical Transient Detection and Estimation Theory

• Derived the maximum-likelihood (ML) detector for a dispersed pulse received by a general array sensor system.

• Showed that the optimal approach to detection in an ML sense is to get time-averaged power from maximum-SNR beams.

• Derived the general ML estimator for an array steering vector when observing a bright point-source.

• Showed that BYU’s approach to array calibration is optimal in an ML sense.

• Derived the ML estimator for the DM of a dispersed pulse when received by a general array sensor.

• Formulated the Fisher information matrix (FIM) for a dispersed pulse model.

• Expanded the above formulation to accommodate a general array sensor.
• Analyzed the CRB for DM as a function of typical receiver parameters.

• Found that the CRB for DM is still relatively low outside the FoV when a strong pulse is received.

• Showed that one should form beams outside the FoV even if they exhibit lower sensitivity.

1.3.3 Radio Frequency Mitigation Algorithms

• Created RFI simulation code that calculates complex PAF voltages and formulates projection operators.

• Developed new algorithm to form spatially broad nulls to mitigate moving RFI while reducing sample estimation error.

• Simulated this new “broad null” algorithm and showed its superiority over existing approaches.

• Studied the effects of processing window sizes on the performance of the broad null algorithm.

• Found a less computationally expensive version of the broad null algorithm.

• Expanded algorithm to cancel multiple RFI sources.

1.4 Dissertation Outline

Chapter 2 introduces a theoretical background to PAF instruments including a signal model, beamforming techniques, and beamformer calibration. The chapter then offers a brief treatment of the basics of astronomical transients including an explanation of the dispersion phenomenon. Lastly, the chapter lays out a foundation for basic interference mitigation techniques for PAFs, including subspace projection.

Chapter 3 outlines the derivations of an astronomical transient ML array-based detector and dispersion-measure estimator. It then offers a derivation of ML estimator for general array steering vectors in the presence of a bright point source.
Chapter 4 presents a simplified astronomical transient model and derives the FIM for it. The CRB for DM is then plotted against various other parameters to illustrate the effects of system configurations and observation scenarios on estimator efficiency.

Chapter 5 discusses the development of our latest and greatest PAF receiver system named the Focal L-Band Array for the Green Bank Telescope (FLAG). A description of FLAG’s various operational modes and capabilities is also given. The chapter then concludes with a summary of commissioning experiments including results from an array calibration and simple science observations.

Chapter 6 deals with RFI mitigation including the introduction of a spatially broad-null-forming algorithm that offers superior cancelation capability compared to the currently accepted approaches. Simulation results of a PAF on the GBT using this and the mainstream algorithms are shown, and comparisons are drawn.
CHAPTER 2. BACKGROUND

2.1 Introduction

In order to understand the work contained in this dissertation, a fair amount of mathematical and theoretical background is required. This chapter aims to fill this requirement by first introducing a general signal model for phased-array feeds (PAFs) and beamforming. It then considers various types of beamformers and explains the array calibration procedure. We then discuss the fundamentals of pulsars, rotating radio transients (RRATs), and fast radio bursts (FRBs) and review established detection and estimation techniques for these signals. A primer on radio-frequency interference (RFI) mitigation techniques, especially subspace projection, is then provided.

2.2 Phased Array Feed Signal Model

Consider a PAF with $M$ arbitrarily distributed elements positioned in the focal plane of a reflector dish. A deep-space point source emission which arrives at the reflector from 2-D arrival angle $\theta_s$ is approximately a plane wave. The resulting received $M \times 1$ vector of sampled instantaneous complex basebanded voltages $x[n]$ for a narrowband frequency channel $k$ is represented by

$$x_k[n] = s_k[n]a_k(\theta_s) + z_k[n],$$  \hspace{1cm} (2.1)

where $s_k[n]$ is the narrowband instantaneous complex basebanded amplitude of the signal of interest (SOI). This is modeled as a temporally white, zero-mean circularly complex Gaussian random process with variance $\sigma^2_{s,k}$. The vector $a_k(\theta_s)$, known as the array steering vector, is the array gain and phase response vector induced by a plane wave arriving from $\theta_s$, and $z_k[n]$ is a temporally white zero-mean circularly complex Gaussian random vector process with spatial covariance.
matrix given by

\[ \mathbf{R}_{z,k} = E \{ \mathbf{z}_k[n] \mathbf{z}_k^H[n] \}, \tag{2.2} \]

where \( E \{ \cdot \} \) is the expectation operator. It should be noted that \( \mathbf{R}_{z,k} \) is not generally diagonal due to array mutual coupling effects, spillover structure, and non-uniform sky noise [66].

When there are multiple, say \( N_s \), plane waves received by the array such as in the case of multiple deep-space point sources, the law of superposition applies and

\[ \mathbf{x}_k[n] = \sum_{l=1}^{N_s} \mathbf{a}_k(\theta_l)s_{l,k}[n] + \mathbf{z}_k[n] = \left[ \begin{array}{c} \mathbf{a}_k(\theta_1) \\ \mathbf{a}_k(\theta_2) \\ \vdots \\ \mathbf{a}_k(\theta_{N_s}) \end{array} \right] \left[ \begin{array}{c} s_{1,k}[n] \\ s_{2,k}[n] \\ \vdots \\ s_{N_s,k}[n] \end{array} \right] + \mathbf{z}[n] \]

\[ = \mathbf{A}_k \mathbf{s}_k[n] + \mathbf{z}_k[n], \tag{2.3} \]

where \( \mathbf{a}_k(\theta_l) \) and \( s_{l,k}[n] \) are the steering vector and complex amplitude of the arriving plane wave corresponding the \( l \)th point source respectively.

By extension, an extended source, or a source that subtends more than the on-sky beamwidth of a reflector antenna, would result in the signal model

\[ \mathbf{x}_k[n] = \int_{\mathbf{R}_\theta} \mathbf{a}_k(\theta)s_k(\theta,n)d\theta + \mathbf{z}[n], \tag{2.4} \]

where \( \mathbf{R}_\theta \) is the SOI’s region of support.

In the presence of \( N_i \) point-source-like interferers, such as from GPS satellite downlinks or ground-based broadcast towers, the multiple point-source signal model becomes

\[ \mathbf{x}_k[n] = \mathbf{A}_{x,k} \mathbf{s}_k[n] + \mathbf{A}_{i,k} \mathbf{i}_k[n] + \mathbf{z}_k[n], \tag{2.5} \]
where \( A_{i,k} \) and \( i_k[n] \) are the RFI steering vectors and plane wave amplitudes respectively and are defined similarly to \( A_{s,k} \) and \( s_k[n] \). In this most general case, under the assumption of statistical independence between \( s_k[n] \), \( i_k[n] \), and \( z_k[n] \), the total spatial array covariance matrix is given by

\[
R_{x,k} = E\{x_k[n]x_k^H[n]\} = A_{s,k}D_{s,k}A_{s,k}^H + A_{i,k}D_{i,k}A_{i,k}^H + R_{z,k},
\]

(2.6)

\[
= R_{s,k} + R_{i,k} + R_{z,k},
\]

(2.7)

where \( D_{s,k} = E\{s_k[n]s_k^H[n]\} \), \( D_{i,k} \) is defined similarly, and \( R_{s,k} \) is the correlation matrix corresponding only to the SOI, and \( R_{i,k} \) and \( R_{z,k} \) are defined similarly. If the wave amplitudes for each SOI are uncorrelated, then \( P_{s,k} \) is a diagonal matrix with time-average power levels as the entries. For radio astronomical sources, due to the vast distances between resolvable points in the deep space object, distinct source points are always uncorrelated due to physical limitations of radio propagation speed. Thus, \( D_{s,k} \) is always diagonal.

We can spatially filter the received voltages through beamforming or

\[
y_k[n] = w_k^Hx_k[n]
\]

(2.8)

\[
= w_k^H(A_{s,k}s_k[n] + A_{i,k}i_k[n] + z_k[n])
\]

\[
= w_k^HA_{s,k}s_k[n] + w_k^HA_{i,k}i_k[n] + w_k^Hz_k[n],
\]

(2.9)

where \( w_k \) is a \( M \times 1 \) vector of complex beamformer weights for the narrowband frequency channel \( k \). The selection of these weights will be discussed in the following section. The time-average beamformed power in a single frequency channel is given by

\[
P_{y,k}[n] = E\{|y_k[n]|^2\}.
\]

(2.10)

Substituting (2.8) into (2.10) yields a convenient equivalent form

\[
P_{y,k}[n] = w_k^H E\{x_k[n]x_k^H[n]\}w_k
\]

\[
= w_k^H R_{x,k} w_k.
\]

(2.11)
Therefore, one is also able to compute the time-averaged beamformed power using spatial covariance matrices. The process in (2.11) is called post-correlation beamforming (PCB) [67].

2.3 Beamformer Weights

In this section, we will consider various ways that one can formulate beamformer weights so as to optimize some desired performance metrics. For example, one can maximize gain, maximize signal-to-noise ratio (SNR), and constrain the beam pattern in certain directions. While these approaches are generally applicable, there are some subtle distinctions between PAFs and aperture arrays that are important in understanding what an ideal weight vector will look like.

Aperture arrays, also known as phased arrays, are sensor systems that operate without the aid of a reflector dish. Consequently, the goal of beamforming is to simply form a far-field pattern by carefully selecting weights that are optimal in some manner depending on the needs of the observer. In a PAF, however, the array is placed at the focus of a large reflector dish, meaning that the goal of beamforming is to change the illumination pattern on the dish in some meaningful fashion. For example, one can attempt to fully utilize the dish area to maximize gain and form a narrow on-sky far-field beam pattern, or one can taper the illumination pattern of the PAF on the dish aperture to better control side lobes and spillover structure. In either scenario, the techniques outlined below can be used.

In this section, it is important to understand that beamformer weights are calculated on a per-narrowband-frequency-channel basis, and thus the weights will have the subscript $k$ to indicate the frequency channel for which they are intended. This is primarily because wider channels induce undesirable phase shifts at the upper and lower limits of the passband, which can cause the beam to off-steer and become misshapened at those edge frequencies. In some cases, one can mitigate this effect through the use of an all-pass, re-phasing finite impulse response (FIR) filter [68], but given the difficulty with calibrating PAFs, it is more convenient to use narrower channels.
2.3.1 Maximum Gain Beamformer

The beamformer weights that result in maximum achievable gain in a particular direction are found by maximizing the beam response in the direction of the SOI, or

\[ w_{k,\text{max-gain}} = \arg\max_w w^H R_{s,k} w \quad \text{such that} \quad \|w\| = 1, \quad (2.12) \]

\[ = \arg\max_w \frac{w^H R_{s,k} w}{w^H w}, \quad (2.13) \]

Through differentiation,

\[ \frac{\partial}{\partial w^H} \left( \frac{w^H R_{s,k} w}{w^H w} \right) = \frac{(w^H w) R_{s,k} w - (w^H R_{s,k} w) w}{(w^H w)^2}, \quad (2.14) \]

which, when set equal to zero, yields

\[ R_{s,k} w_{k,\text{max-gain}} = \frac{w_{k,\text{max-gain}}^H R_{s,k} w_{k,\text{max-gain}}}{w_{k,\text{max-gain}}^H w_{k,\text{max-gain}}} w_{k,\text{max-gain}}. \quad (2.15) \]

In the point-source case, \( R_{s,k} = \sigma_s^2 a_k(\theta_s) a_k^H(\theta_s) \), and

\[ w_{k,\text{max-gain}} = \frac{w_{k,\text{max-gain}}^H w_{k,\text{max-gain}}}{w_{k,\text{max-gain}}^H a_k(\theta_s)} a_k(\theta_s), \]

\[ = \alpha a_k(\theta_s), \quad (2.16) \]

where \( \alpha \) is the product of all the multiplicative scalars, which can be chosen so as to constrain the beam peak response level. For example, to obtain a unit beam response in the beam direction, we let

\[ w_{k,\text{max-gain}} = \frac{1}{a_k^H(\theta_s) a_k(\theta_s)} a_k(\theta_s). \quad (2.17) \]

These weights will result in maximum array gain and are sometimes referred to as the conjugate field match (CFM) beamformer [18]. However, in the presence of spatially colored noise, these weights do not result in maximum SNR.
2.3.2 Maximum SNR Beamformer

The maximum-SNR beamformer is the set of weights that maximizes the post-beamforming SNR [67], [68], or,

\[
\mathbf{w}_{k,\text{max-SNR}} = \arg\max_w \frac{\mathbf{w}^H \mathbf{R}_{z,k} \mathbf{w}}{\mathbf{w}^H \mathbf{R}_{z,k} \mathbf{w}},
\]  

(2.18)

Similar to the maximum-gain case, differentiation of this ratio leads to the generalized eigenvalue problem

\[
\mathbf{R}_{s,k} \mathbf{w}_{k,\text{max-SNR}} = \lambda_{\text{max}} \mathbf{R}_{z,k} \mathbf{w}_{k,\text{max-SNR}},
\]  

(2.19)

where \(\lambda_{\text{max}}\) is the maximum eigenvalue of \(\mathbf{R}_{z,k}^{-1} \mathbf{R}_{s,k}\).

In the point source case, \(\mathbf{R}_{s,k} = \sigma_s^2 \mathbf{\alpha}_k(\theta_s) \mathbf{\alpha}_k(\theta_s)\), and

\[
\mathbf{w}_{k,\text{max-SNR}} = \frac{\mathbf{w}_{k,\text{max-SNR}}^H \mathbf{R}_{z,k} \mathbf{w}_{k,\text{max-SNR}}}{\mathbf{w}_{k,\text{max-SNR}}^H \mathbf{R}_{z,k} \mathbf{R}_{z,k}^{-1} \mathbf{\alpha}_k(\theta_s)} \mathbf{R}_{z,k}^{-1} \mathbf{\alpha}_k(\theta_s)
\]

\[
= \alpha \mathbf{R}_{z,k}^{-1} \mathbf{\alpha}_k(\theta_s).
\]  

(2.20)

As with the maximum-gain beamformer, the scale factor \(\alpha\) can be selected to elicit a desired formed beam peak response in the beam direction. It should be noted that if the noise is spatially white (i.e., \(\mathbf{R}_{z,k} = a \mathbf{I}\), for arbitrary scalar \(a\)), then the maximum-SNR and gain-maximizing weights are identical.

2.3.3 Linear Constraint Minimum Variance Beamformer

More control over the beam pattern can be achieved by introducing constraints on the pattern response and minimizing the total beamformed power subject to said constraints. This results in the linear constraint minimum variance (LCMV) beamformer, which is formulated as

\[
\mathbf{w}_{k,\text{LCMV}} = \arg\min_w \mathbf{w}^H \mathbf{R}_{z,k} \mathbf{w}, \text{ subject to } \mathbf{C}^H \mathbf{w} = \mathbf{f},
\]  

(2.21)
where \( C \) is a \( P \times M \) matrix of \( P \) array steering vectors corresponding to angular directions to be constrained, and \( f \) is a \( P \times 1 \) array of desired response pattern levels in those directions. The closed form solution to this optimization problem is given by [54]

\[
\mathbf{w}_{k, \text{LCMV}} = \mathbf{R}^{-1}_{z,k} \mathbf{C} \left( \mathbf{C}^H \mathbf{R}^{-1}_{z,k} \mathbf{C} \right)^{-1} f. \tag{2.22}
\]

When \( C = \mathbf{a}_k(\theta_s) \) and \( f = 1 \), this yields the minimum-variance distortionless response (MVDR) beamformer, or

\[
\mathbf{w}_{k, \text{MDVR}} = \frac{1}{\mathbf{a}^H_k(\theta_s) \mathbf{R}_{z,k} \mathbf{a}_k(\theta_s)} \mathbf{R}^{-1}_{z,k} \mathbf{a}_k(\theta_s) \\
= \alpha \mathbf{R}^{-1}_{z,k} \mathbf{a}_k(\theta_s),
\]

which is the same result as (2.20).

The constraint points are often placed around the desired 3-dB contour and nearby sidelobes to provide reliably stable beam patterns [31], [69], [70]. An arbitrary number of constraints can be incorporated into the optimization, but a large number of constraint points reduce the degrees of freedom that can be used for variance minimization and can result in lower SNR.

### 2.4 Array Feed Calibration

In all of our beamformer weight formulations, we have assumed knowledge of the array steering vector in some fashion, such as with \( \mathbf{a}_k(\theta_s) \) or other steering vectors in the constraint matrix \( C \) from the LCMV formulation. Generally speaking, the array steering vectors are not known \textit{a priori} due to unknown signal path gains and phase delays, structural scattering, mutual coupling, focus offset, and receiver drift. Consequently, one must estimate the steering vectors for each desired beam and constraint direction in order to formulate beam weights. This process of estimating steering vectors per angle of arrival is known as array calibration.

To calibrate a radio astronomical array, we observe a bright point source such that our signal model is represented by (2.1), where \( \theta_s \) is the relative angle between the source and the dish pointing vector. One then estimates the steering vector through extracting the dominant eigenvector...
Figure 2.1: PAFs are calibrated by steering the dish across a strong point source and estimating the corresponding steering vector vectors for each pointing. The red dots indicate an “on” pointing, and the blue dots indicate an “off” pointing.

of the generalized eigenvalue problem

\[ \hat{R}_{\text{on},k}(\theta_s)v_k = \lambda_{\text{max},k} \hat{R}_{\text{off},k}v_k, \]

(2.23)

where \( v_k \) is an eigenvector corresponding the largest eigenvalue \( \lambda_{\text{max},k} \), \( \hat{R}_{\text{on},k}(\theta_s) \) is the sample spatial correlation matrix when the bright point source is positioned at the angle \( \theta_s \), and \( \hat{R}_{\text{off},k} \) is a noise-only sample correlation matrix. In some scenarios, the structure of the “off” correlation matrix varies with elevation angle, so it is desirable to obtain an “off” for each surveyed elevation angle. The estimated array steering vector is then given by

\[ \hat{a}_k(\theta) = \sqrt{\lambda_{\text{max},k}} \hat{R}_{\text{off}}v_k. \]

(2.24)

A simple procedure to estimate several steering vectors in the field of view is to steer the dish in a serpentine pattern across the bright point source with a short dwell time at each angle of interest, as shown in Figure 2.1. This results in a convenient rectangular grid, which facilitates array characterization and imaging.
2.5 Fundamentals of Astronomical Transients

Astronomical transients can be characterized as broadband bursts of energy that originate from a highly localized region and are dispersed by travel through a turbulent plasma known as the interstellar medium (ISM) [1]. This dispersion is quantified to a single number called the dispersion measure (DM), which has units of cm\(^{-3}\) pc, where pc stands for parsecs. Objects that are more distant undergo more dispersion since they propagate, in total, through more charged particles. Thus, a measured DM can be used as a rough estimate of distance. The relative time delay between two frequencies for a given DM is

\[ \Delta t = \mathcal{D}(f_1^{-2} - f_2^{-2})D, \]  

where \( \mathcal{D} = 4.1488 \times 10^6 \text{ MHz}^2 \text{ pc}^{-1} \text{ cm}^3 \text{ ms} \) is called the dispersion constant, \( f_1 \) and \( f_2 \) are the two frequencies of interest measured in MHz, and \( D \) is the DM. The function \( h_k(D,m,t_0,T) \) is the pulse shape model, which is discussed further in Chapter 4. Under these assumptions, the astronomical transient signal model for a single pulse is given by

\[ x_k[mN+n] = h_k(D,m,t_0,T)s_k[mN+n]a_k(\theta_s) + z_k[mN+n], \]  

where \( m, 1 \leq m \leq M \) is a short-time window (STW) index during which we assume the pulse amplitude is approximately constant, \( N \) is the length of said STI window, \( D \) is the dispersion measure, \( t_0 \) is the pulse start time in seconds, and \( T \) is the pulse width in seconds.

Generally, the dispersion measure \( D \) of an astronomical transient is unknown, so pulse detection typically involves a series of “de-dispersion” trials and looking for peaks. This is usually done incoherently across power-accumulated channelized data for each beam or

\[ \tau = \max_D \sum_{k=0}^{K} [y_k[M_D(k)]]^2, \]  

where \( \tau \) is the detection statistic to test against a threshold, \( K \) is the number of frequency channels in the digital signal processor, \( y_k[j] \) is the accumulated power in the \( k \)th frequency channel in the \( j \)th short-time integration (STI) window, and \( M_D(k) \) is a mapping from frequency channel \( k \) to the
time delay for that frequency channel induced by a DM of $D$, i.e.,

$$M_D(k) = \left\lfloor \mathcal{D}(f_0^{-2} - f_k^{-2})D \right\rfloor,$$

(2.28)

where $f_k$ is the center frequency of the $k$th frequency channel, and $\lfloor \cdot \rfloor$ is the rounding operator.

While pulse shapes can vary [1], they are commonly Gaussian and can be modeled as

$$h_k(D, m, t_0, T) = \exp \left\{ -\frac{(mT_s - (t_0 - 2T - M_D(k)))^2}{4T} \right\},$$

(2.29)

where $T_s$ is the STI length, the center of the Gaussian is $t_0 - 2T$, and its variance is $2T$.

The three major types of astronomical bursts that will be covered here are pulsars, RRATs, and FRBs. All of these experience the dispersive channel of the ISM as described above.

Pulsars are signals that generate highly periodic pulses with periods ranging from several seconds to milliseconds [1]. They are believed to be rotating neutron stars that emit a broadband beam of energy much like a lighthouse. A pulse is observed when this beam points towards a radio telescope on the Earth. Most surveys of pulsars attempt to improve their sensitivity by first applying incoherent de-dispersion as shown in (2.27) and then “folding” the pulses by summing harmonics of the pulse frequency over a long observation time window.

RRATs, much like pulsars, are periodic, but they will frequently miss several periods [71], [72], rendering the folding technique less effective if not completely ineffective. In lieu of folding, RRATs are surveyed using single-pulse searches [73]–[75].

FRBs, on the other hand, are single, relatively high energy one-time bursts [76] that usually last a few milliseconds [2], [77] and have pulse shapes that suggest scattering in the ISM [78], [79]. Only 23 FRBs that have been documented to date [4]–[17], and their origin is still unknown. Many have offered possible explanations for the origin of FRBs [80]–[83], but without more documented cases, this likely will remain an unanswered question.
2.6 Detection of Astronomical Transients

As stated previously, the currently accepted approach to astronomical transient single-pulse detection is to incoherently sum in frequency and time for various DM trials, or

\[
\tau = \max_{D, T, t_0} \sum_{t=t_0}^{t_0+T} \sum_{k=0}^{K} y_k [t - M_D(k)],
\]

(2.30)

where we have also introduced the additional maximizing criteria of start time \( t_0 \) and pulse width \( T \), since these are not typically known \textit{a priori}. In this approach, detection requires a search over a series of DM, pulse start time, and pulse width trials. The selection of trial DMs can be found using [1]

\[
DM_i = 1.205 \times 10^{-7}\text{cm}^{-3}\text{pc} (i-1) t_{\text{samp}} \left( \frac{f^3}{\Delta f} \right),
\]

(2.31)

where \( t_{\text{samp}} \) is the sample period in milliseconds, \( f \) is the center on-sky frequency in MHz, and \( \Delta f \) is the bandwidth in MHz of each frequency bin. This results in a very large number of trial DMs, each of which require \( N_\Delta K N \) floating point operations to perform \( N_\Delta \) de-dispersion trials for each window of \( N \) time samples.

Alternatively, one can use “tree-dedispersion,” which is a divide-and-conquer algorithm for de-dispersion and reduces the floating point operations to \( KN \log_2 K \) [84]. This approach, however, is not amenable to parallelization, which has resulted in the emergence of several “brute-force” algorithms that employ parallel processing hardware such as FPGAs and graphics processing units (GPUs) [85]–[87]. The most recent approach to de-dispersion uses the fast dispersion measure transform (FDMT), which transforms a data set from a frequency-time domain into a DM-time one and has a computational complexity of \( \max\{N_\Delta N \log_2 K, 2KN\} \) [22].

After de-dispersion, one must also account for the pulse width, which, when large relative to the sampling period, makes detection difficult. To mitigate this, it is common to add neighboring time samples as seen in the limits of the summation in (2.30). This process is sometimes referred to as “boxcar smoothing” or using a “boxcar filter” [88]–[90], and ultimately makes the single-pulse detection problem an exercise in sub-optimal matched filter identification [1].
2.7 Estimation of Astronomical Transient Parameters

There are numerous parameters of interest to astronomers in astronomical transients. For example, precise times of arrival (TOA) referenced to center of the solar system (called the barycentric frame, which is used to remove year-scale artifacts of Doppler shifting or time delays induced by the Earth’s orbital rotation) yield valuable information about a pulsar, its location, the ISM, and massive bodies near the propagation path. The process of finding these TOAs is known as pulsar timing.

The current standard in pulsar timing, or the estimation of transient parameters, is currently found in the software package TEMPO2 [47], [48]. TEMPO2 uses observations of a pulsar over the course of months to years and fits the data to a timing model to estimate the transient’s parameters. The accuracy of these estimates is then evaluated through residual analysis, or the difference between the fitted model and the data [49], [51], [52].

2.8 Radio-Frequency Interference Mitigation Methods

There are various forms of radio-frequency interference (RFI) that can plague an astronomical survey. These include but are not limited to impulsive RFI (lightning strikes, pulsed radars, microwave ovens), narrowband satellite communication downlinks (GPS, satellite TV, satellite Internet) and broadcasting towers (radio stations, cell phone towers), and unshielded electronic broadband noise (network switches, computers, digital cameras).

Depending on the application, RFI mitigation can take place in the form of prevention, detection and discarding in time and frequency, and time-domain-based filtering and subtraction. The spatial diversity inherent in phased arrays adds adaptive spatial filtering and beamforming to the RFI-mitigation toolbox.

One way to mitigate the RFI is to simply include the spatial correlations induced by the interference in $\mathbf{R}_x$ when using the LCMV (or a related minimum variance) beamformer approach. Thus, the beamformer weights naturally place a null in the RFI direction. Unfortunately, minimum variance techniques, while effective in partially attenuating RFI, are typically not sufficient methods for RFI mitigation in astronomy applications. Such approaches can only attenuate the RFI signal component down to a level close to the noise floor, but astronomical signals are typically
much weaker and fall well below the noise floor, resulting in poor post-cancelation signal-to-interference ratios (SIRs) and, consequently, poor detection performance. Consequently, we favor the zero-forcing properties of orthogonal projection approaches in radio astronomy.

2.8.1 Orthogonal Projections

In projection-based RFI mitigation, the goal is to get an estimate of the RFI subspace $\mathbf{U}$, or the space spanned by the steering vectors of the impinging RFI on a per-narrowband-channel basis. We can then form the projection matrix $\mathbf{P} = \mathbf{I} - \mathbf{U}(\mathbf{U}^H\mathbf{U})^{-1}\mathbf{U}^H$, which projects $\mathbf{x}[n]$ onto the orthogonal complement of $\mathbf{U}$ such that

$$
\mathbf{P}\mathbf{x}[n] = s[n]\mathbf{P}\mathbf{a}(\theta_s) + i[n]\mathbf{P}\mathbf{a}(\theta_i) + \mathbf{P}\mathbf{\eta}[n] \quad (2.32)
$$

where, for simplicity, we have dropped the subscript $k$ with the understanding that this process is to be applied individually to all affected frequency channels. Equivalently, the projection matrix can be applied on spatial covariance matrices such that

$$
\mathbf{P}\mathbf{R}_x\mathbf{P}^H = \mathbf{PR}_s\mathbf{P}^H + \mathbf{PR}_i\mathbf{P}^H + \mathbf{PR}_\mathbf{\eta}\mathbf{P}^H \\
\approx \mathbf{R}_s + \mathbf{R}_\mathbf{\eta}. \quad (2.33)
$$

This approach may cause a undesirable scaling and rotation of the signal subspace ($\mathbf{P}\mathbf{a}(\theta_s) \approx \mathbf{a}(\theta_s)$, not equal) so the beam pattern becomes partially distorted. On average, the unbiased RFI-free covariance matrix is recoverable [56] by computing several successive projection matrices $\mathbf{P}_l$, with $1 \leq l \leq L$, and corresponding “cleaned” covariances $\hat{\mathbf{R}}_{x,l} = \mathbf{P}_l\mathbf{R}_{x,l}\mathbf{P}_l^H$ according to

$$
\hat{\mathbf{R}}_{x,\text{unbiased}} = \text{unvec} \left\{ \mathbf{C}^{-1} \text{vec} \left( \frac{1}{L} \sum_{l=1}^{L} \hat{\mathbf{R}}_{x,l} \right) \right\}, \quad (2.34)
$$
where
\[ C = \frac{1}{L} \sum_{l=1}^{L} P^H_l \otimes P_l. \] (2.35)

This bias correction relies on \( C \) being full-rank, which depends on sufficient motion of the interference. However, \( C \) can easily be ill-conditioned so a pseudo-inverse is used where all singular values of \( C \) that are less than \( \text{tr}(C)/M^2 \) are set to zero [57], [58].

Unfortunately, it is practically impossible to know \( a(\theta_i) \) \textit{a priori} due to uncalibrated regions of the array such as in deep sidelobes of a large reflector, making in situ estimation of the RFI steering vectors necessary. The rank-one estimate of the RFI steering vector for a short-time window in which the RFI is stationary is found by solving the generalized eigenvalue problem
\[
\hat{\mathbf{R}}_s \hat{\mathbf{a}} = \lambda_{\text{max}} \hat{\mathbf{R}}_\eta \hat{\mathbf{a}},
\] (2.36)
where \( \hat{\mathbf{R}}_s = \frac{1}{N} \sum_{n=0}^{N} \mathbf{x}[n] \mathbf{x}^H[n] \), and the dominant unit-length normalized eigenvector is our estimate of \( a(\theta_i) \). The projection matrix is then given by
\[
\mathbf{P} = \mathbf{I} - \hat{\mathbf{a}} \hat{\mathbf{a}}^H.
\] (2.37)

The matrix \( \mathbf{P} \) can place nulls by modifying the beamformer weights such that
\[
y_k[n] = \mathbf{w}_k^H \mathbf{P}_k^H \mathbf{x}_k[n],
\]
\[
= (\mathbf{P}_k \mathbf{w}_k)^H \mathbf{x}_k[n],
\]
\[
= \tilde{w}_k^H \mathbf{x}_k[n].
\] (2.38)

The projection matrix can also be used in post-correlation beamforming and interferometric imaging [57], [58] by
\[
\hat{\mathbf{R}}_x = \mathbf{P} \hat{\mathbf{R}}_x \mathbf{P}.
\] (2.39)

This approach is known as subspace projection.
CHAPTER 3. MAXIMUM LIKELIHOOD DETECTION AND ESTIMATION FOR PHASED ARRAY FEEDS

3.1 Introduction

Since phased array feeds (PAFs) are a relatively new instrument in the radio astronomy community, there has not been a significant amount of detection and estimation theory development specific to these unique array systems. For example, many beamforming techniques require accurate knowledge of the array steering vector $a_k(\theta)$. However, this vector is not easily modeled for a PAF due to imperfections in the optics, scattering off the feed arms, diffraction around the edge of the dish, non-uniform sky noise, dish illumination pattern spillover structure, and mutual coupling. Consequently, this vector must estimated in situ before any kind of beamforming can take place. The currently practiced methods for estimating $a_k(\theta)$ have been successful in observation campaigns using PAFs, but these approaches arose out of raw intuition and experience rather than a formal derivation.

Additionally, there does not appear to be any formal detection and estimation derivations with respect to astronomical transients such as pulsars and fast radio bursts (FRBs). For example, the current approach to parameter estimation is to fit a long series of data to a model and attempt to minimize residuals. While this approach has produced good results, it is only optimal in a least-squares sense and could be improved.

This chapter aims to fill this void by providing derivations for optimal detectors and parameter estimators in a maximum-likelihood (ML) sense for PAF-based radio astronomical instruments. First, the signal model from Chapter 2 is restated, and the ML criterion for detectors and estimators is provided. Then the ML estimator for array steering vectors is derived, followed by a derivation of the ML detector for a general astronomical transient using an array system.
3.2 Maximum Likelihood Framework

Recall the signal model from (6.1), which is repeated here for convenience:

\[ x_k[n] = s_k[n]a_k(\theta_s) + z_k[n]. \] (3.1)

Generally the vectors \( x_k[n] \) will be used for most of the derivations in this chapter. The ML detector criterion is given by

\[ \hat{m} = \arg\max_m p(x_1[1], x_1[2] \cdots x_K[N]|H_m), \] (3.2)

where \( H_m \) is the \( m \)th hypothesis, and \( p(x_k[n]|H_m) \) is the joint conditional probability density function (PDF) for \( N \) array sample vectors over \( K \) frequency channels conditioned on \( H_m \). We assume independence between time and frequency samples due to the thermal blackbody radiation origins of radio astronomical systems and the narrowband frequency channelized processing. Both SOI and noise appear spectrally white across adjacent channels, resulting in

\[ \hat{m} = \arg\max_m \prod_{k=1}^K \prod_{n=1}^N p(x_k[n]|H_m) \] (3.3)

\[ = \arg\max_m L(H_m), \] (3.4)

where \( L(H_m) \) is known as the likelihood function. Thus an ML detector decides a given hypothesis \( H_{m'} \) when its likelihood function \( L(H_{m'}) \) is the largest, or

\[ L(H_{m'}) > L(H_m), \text{ for all } m \neq m'. \] (3.5)

It is often convenient (e.g., for exponential PDF families) to examine the logarithm of the likelihood function, which is referred to as the log-likelihood function and is given by

\[ \Lambda(H_m) = \ln L(H_m) \] (3.6)

\[ = \sum_{k=1}^K \sum_{n=1}^N \ln p(x_k[n]|H_m). \] (3.7)
Since the natural logarithm is a monotonic function, \( \Lambda(H_m) \) can be used interchangeably with \( L(H_m) \) in the optimization, and \( \hat{m} \) remains the same, or

\[
\hat{m} = \arg \max_m \Lambda(H_m).
\] (3.8)

When estimating a parameter \( \theta \), we instead find the likelihood function conditioned on the parameter instead of a discrete hypothesis as in (3.8), so

\[
\hat{\theta} = \arg \max_{\theta} \Lambda(\theta)
\] (3.9)
\[
= \arg \max_{\theta} \sum_{k=1}^{K} \sum_{n=1}^{N} \ln p(x_k[n]|\theta).
\] (3.10)

### 3.2.1 Useful Theorems

Before moving forward with the derivations, there are a number of linear algebraic theorems and lemmas that are critical to simplify the resulting detectors and estimators.

**Matrix Determinant Lemma** [91]

\[
|A + uv^H| = (1 + v^H A^{-1} u)|A|.
\] (3.11)

**Generalized Matrix Determinant Lemma** [91]

\[
|A + UCV^T| = |C^{-1} + V^T A^{-1} U||C||A|.
\] (3.12)

**Sherman-Morrison Formula** [92]

\[
(A + uv^H)^{-1} = A^{-1} - \frac{A^{-1}uv^H A^{-1}}{1 + v^H A^{-1} u}.
\] (3.13)

**Woodbury Matrix Identity** [93]

\[
(A + UCV)^{-1} = A^{-1} - A^{-1} U(C^{-1} + VA^{-1} U)^{-1} VA^{-1}.
\] (3.14)
These identities and formulas will be used throughout the derivations that follow to simplify the process.

3.3 Array Steering Vector Estimator

Generally speaking, the array steering vector $a_k(\theta)$ varies with frequency channel $k$, and one must thus form estimates on a per-frequency-channel basis. We note here that for steering vector estimation, data samples are not strictly independent across frequency channels because $a_k(\theta)$ varies relatively smoothly with frequency. For simplicity, however, we choose to consider each channel steering vector estimate as uncoupled with neighboring channels. For this reason, the subscript $k$ will be dropped in this section with the understanding that this process must be repeated for each frequency channel $k$, $1 \leq k \leq K$. In this case, the ML criterion is given by

$$
\hat{a}(\theta) = \arg\max_{a(\theta)} \Lambda(a(\theta))
$$

$$
= \arg\max_{a(\theta)} \sum_{n=1}^{N} \ln p(x[n]|a(\theta)).
$$

Note that the summation over frequency channels as seen in (3.10) is absent since $a(\theta)$ is estimated with data from a single frequency channel.

Recall that $x[n]$ is modeled as a zero-mean circularly complex Gaussian random process with covariance $R_x = \sigma_a^2 a(\theta)a^H(\theta) + R_z$. Therefore, the conditional PDF $p(x[n]|a(\theta))$ is given by

$$
p(x[n]|a(\theta)) = \frac{1}{\pi^{M/2}|\sigma_a^2 a(\theta)a^H(\theta) + R_z|} e^{-x^H[n](\sigma_a^2 a(\theta)a^H(\theta) + R_z)^{-1}x[n]}.
$$

The log-likelihood function, after dropping all constant additive terms to do not affect $\hat{a}(\theta)$, is then

$$
\Lambda(a(\theta)) = -N \ln|\sigma_a^2 a(\theta)a^H(\theta) + R_z| - \sum_{n=1}^{N} x^H[n](\sigma_a^2 a(\theta)a^H(\theta) + R_z)^{-1}x^H[n].
$$
Therefore, we then use the matrix determinant lemma from (3.11), and again drop additive constant terms to obtain

\[ \Lambda(a(\theta)) = -N \ln(1 + \sigma_s^2 a^H(\theta) R_z^{-1} a(\theta)) - \sum_{n=1}^{N} x^H[n] \left( \sigma_s^2 a(\theta) a^H(\theta) + R_z \right)^{-1} x^H[n]. \]  

(3.19)

Further the Sherman-Morrison formula from (3.13) simplifies this to

\[ \Lambda(a(\theta)) = -N \ln(1 + \sigma_s^2 a^H(\theta) R_z^{-1} a(\theta)) + \sum_{n=1}^{N} \left( \frac{\sigma_s^2 x^H[n] R_z^{-1} a(\theta) a^H(\theta) R_z^{-1} x[n]}{1 + \sigma_s^2 a^H(\theta) R_z^{-1} a(\theta)} \right). \]

(3.20)

Since \( x^H[n] R_z^{-1} a(\theta) \) is a scalar,

\[ x^H[n] R_z^{-1} a(\theta) a^H(\theta) R_z^{-1} x[n] = |x^H[n] R_z^{-1} a(\theta)|^2 \]

\[ = a^H(\theta) R_z^{-1} x[n] x^H[n] R_z^{-1} a(\theta). \]

Therefore,

\[ \Lambda(a(\theta)) = -N \ln(1 + \sigma_s^2 a^H(\theta) R_z^{-1} a(\theta)) + \frac{\sigma_s^2 a^H(\theta) R_z^{-1} \left( \sum_{n=1}^{N} x[n] x^H[n] \right) R_z^{-1} a(\theta)}{1 + \sigma_s^2 a^H(\theta) R_z^{-1} a(\theta)} \]

\[ = -N \ln(1 + \sigma_s^2 a^H(\theta) R_z^{-1} a(\theta)) + \frac{\sigma_s^2 a^H(\theta) R_z^{-1} \hat{R}_x R_z^{-1} a(\theta)}{1 + \sigma_s^2 a^H(\theta) R_z^{-1} a(\theta)}, \]

(3.21)

where \( \hat{R}_x \) is the sample spatial covariance matrix.

A necessary condition for \( \hat{a}(\theta) \) to maximize the log-likelihood function is

\[ \frac{\partial \Lambda(a(\theta))}{\partial a(\theta)} \bigg|_{a(\theta)=\hat{a}(\theta)} = 0. \]

(3.22)

The partial derivative of the log-likelihood function is

\[ \frac{\partial \Lambda(a(\theta))}{\partial a(\theta)} = -\frac{N \sigma_s^2 R_z^{-1} a(\theta)}{1 + \sigma_s^2 a^H(\theta) R_z^{-1} a(\theta)} + \frac{N \sigma_s^2 (1 + \sigma_s^2 a^H(\theta) R_z^{-1} a(\theta)) \hat{R}_x R_z^{-1} a(\theta)}{(1 + \sigma_s^2 a^H(\theta) R_z^{-1} a(\theta))^2} \]

\[ - \frac{N \sigma_s^2 a^H(\theta) R_z^{-1} \hat{R}_x R_z^{-1} a(\theta) R_z^{-1} a(\theta)}{(1 + \sigma_s^2 a^H(\theta) R_z^{-1} a(\theta))^2}. \]

(3.23)
Setting the derivative equal to zero, \( a(\theta) \) becomes \( \hat{a}(\theta) \) and

\[
R_x^{-1} \hat{R}_x R_x^{-1} \hat{a}(\theta) = \left[ \frac{\sigma^2 \hat{a}^H(\theta) R_x^{-1} \hat{R}_x R_x^{-1} \hat{a}(\theta)}{1 + \sigma^2 \hat{a}^H(\theta) R_x^{-1} \hat{a}(\theta)} - 1 \right] R_x^{-1} \hat{a}(\theta) = \lambda R_x^{-1} \hat{a}(\theta). \tag{3.24}
\]

Let \( \tilde{a}(\theta) = R_x^{-1} \hat{a}(\theta) \), which results in

\[
\hat{R}_x \tilde{a}(\theta) = \lambda R_x \tilde{a}(\theta). \tag{3.25}
\]

This is a generalized eigenvalue problem, where \( \tilde{a}(\theta) \) is the eigenvector corresponding to the largest magnitude eigenvalue of \( R_x^{-1} \hat{R}_x \). Thus the ML estimated steering vector is

\[
\hat{a}(\theta) = \sqrt{\lambda} R_x \tilde{a}(\theta), \tag{3.26}
\]

where the \( \sqrt{\lambda} \) term scales the unit-norm eigenvector such that the gain terms of \( a(\theta) \) are included in the estimate \( \hat{a} \). This happens to be the practiced approach to PAF calibration introduced in Section 2.4, indicating that the existing methods for array calibration are optimal in an ML sense at least for isolated frequency channels.

### 3.4 Transient Detection in Element Space

In a simple transient detection problem, there are two hypotheses given by

\[
H_0 : \text{No Transient Present} \nonumber
\]
\[
H_1 : \text{Transient Present}. \nonumber
\]

The log-likelihood function is given by (3.7), where, under \( H_1 \), \( x_k[n] \) incorporates a pulse shape model such that

\[
x_k[n] = P_k[n] s_k[n] a_k(\theta) + z_k[n], \tag{3.27}
\]
where $P_k[n]$ is the pulse’s time-frequency shape constrained in our model such that $0 \leq P_k[n] \leq 1$

When the pulse is not present, or $H_0$ is true, $P_k[n] = 0$, and

$$x_k[n] = z_k[n]. \quad (3.28)$$

Using this model the spatial covariance matrix for the received vector $x_k[n]$ under $H_1$ becomes

$$R_{x,k}[n] = E\{x_k[n]x_k^H[n]\}$$
$$= P_k^2[n]E\{|s_k[n]|^2\}a_k(\theta)a_k^H(\theta) + E\{z_k[n]z_k^H[n]\}$$
$$= P_k^2[n]\sigma^2_{s,k}a_k(\theta)a_k^H(\theta) + R_{z,k}.$$

Therefore,

$$p(x_k[n]|H_0) = \frac{1}{\pi^M |R_{z,k}|} e^{-x_k^H[n]R_{z,k}^{-1}x_k[n]}$$
$$p(x_k[n]|H_1) = \frac{1}{\pi^M [P_k^2[n]\sigma^2_{s,k}a_k(\theta)a_k^H(\theta) + R_{z,k}]} e^{-x_k^H[n](P_k^2[n]\sigma^2_{s,k}a_k(\theta)a_k^H(\theta) + R_{z,k})^{-1}x_k[n]}.$$

The hypothesis $H_1$ is decided when

$$\sum_{k=1}^K \sum_{n=1}^N \ln p(x_k[n]|H_1) > \sum_{k=1}^K \sum_{n=1}^N \ln p(x_k[n]|H_0). \quad (3.29)$$

Thus, we choose $H_1$ (transient present) when

$$\sum_{k=1}^K \sum_{n=1}^N \ln |R_{z,k}| + x_k^H[n]R_{z,k}^{-1}x_k[n]$$
$$> \sum_{k=1}^K \sum_{n=1}^N \ln [P_k^2[n]\sigma^2_{s,k}a_k(\theta)a_k^H(\theta) + R_{z,k}] + x_k^H[n](P_k^2[n]\sigma^2_{s,k}a_k(\theta)a_k^H(\theta) + R_{z,k})^{-1}x_k[n]. \quad (3.30)$$

31
After some additional simplification,
\[
\sum_{k=1}^{K} \sum_{n=1}^{N} x_k^H[n](\mathbf{R}_{z,k}^{-1} - (P_k^2[n]\sigma_{s,k}^2\mathbf{a}_k^H(\theta)\mathbf{a}_k^H(\theta) + \mathbf{R}_{z,k})^{-1})x_k[n]
\]
\[
> \sum_{k=1}^{K} \sum_{n=1}^{N} \ln |P_k^2[n]\sigma_{s,k}^2\mathbf{a}_k^H(\theta)\mathbf{a}_k^H(\theta) + \mathbf{R}_{z,k}| - \ln |\mathbf{R}_{z,k}|.
\]
(3.31)

The matrix determinant lemma from (3.11) is then used to simplify this further, such that
\[
\ln |P_k^2[n]\sigma_{s,k}^2\mathbf{a}_k^H(\theta)\mathbf{a}_k^H(\theta) + \mathbf{R}_{z,k}| - \ln |\mathbf{R}_{z,k}| = \ln(1 + P_k^2[n]\sigma_{s,k}^2\mathbf{a}_k^H(\theta)\mathbf{R}_{z,k}^{-1}\mathbf{a}_k(\theta)).
\]
(3.32)

Applying the Sherman-Morrison formula from (3.13),
\[
\mathbf{R}_{z,k}^{-1} - (P_k^2[n]\sigma_{s,k}^2\mathbf{a}_k^H(\theta)\mathbf{a}_k^H(\theta) + \mathbf{R}_{z,k})^{-1} = \frac{P_k^2[n]\sigma_{s,k}^2 \mathbf{R}_{z,k}^{-1}\mathbf{a}_k(\theta)\mathbf{a}_k^H(\theta) \mathbf{R}_{z,k}^{-1}\mathbf{a}_k(\theta)}{1 + P_k^2[n]\sigma_{s,k}^2\mathbf{a}_k^H(\theta)\mathbf{R}_{z,k}^{-1}\mathbf{a}_k(\theta)}.
\]
(3.33)

The terms in (3.32) and (3.33) are then substituted into (3.31) to obtain the detector (i.e., we choose \(H_1\) when)
\[
\sum_{k=1}^{K} \sum_{n=1}^{N} \frac{P_k^2[n]\sigma_{s,k}^2 |\mathbf{a}_k^H(\theta)\mathbf{R}_{z,k}^{-1}x_k[n]|^2}{1 + P_k^2[n]\sigma_{s,k}^2\mathbf{a}_k^H(\theta)\mathbf{R}_{z,k}^{-1}\mathbf{a}_k(\theta)} > \sum_{k=1}^{K} \sum_{n=1}^{N} \ln(1 + P_k^2[n]\sigma_{s,k}^2\mathbf{a}_k^H(\theta)\mathbf{R}_{z,k}^{-1}\mathbf{a}_k(\theta)).
\]
(3.34)

This result has many moving parts to it, and implementing such a detector is complicated. To help further simplify this detector formulation, the following sections will explore how (3.34) simplifies under varying assumptions and observation scenarios.

### 3.4.1 Spatially White Noise

Consider the scenario in which the array noise is spatially white, or
\[
\mathbf{R}_{z,k} = \sigma_{z,k}^2 \mathbf{I},
\]
(3.35)
where $\sigma^2_{z,k}$ is the noise variance for the $k$th frequency channel. Given (3.35), (3.34) becomes

$$
\frac{K}{k=1} \sum_{n=1}^{N} \frac{P_k^2[n] \sigma^2_{z,k} |a_k^H(\theta)x_k[n]|^2}{\sigma^2_{z,k} \left(1 + P_k^2[n] \sigma^2_{z,k} \sigma^2_{x,k} / \sigma^2_{z,k} \right)} > \frac{K}{k=1} \sum_{n=1}^{N} \ln \left(1 + P_k^2[n] \frac{\sigma^2_{x,k} a_k^H(\theta) a_k(\theta)}{\sigma^2_{z,k}} \right)
$$

(3.36)

$$
\sum_{k=1}^{K} \sum_{n=1}^{N} \frac{\text{SNR}_k(\theta, n) |\tilde{a}_k^H(\theta)x_k[n]|^2}{\sigma^2_{z,k} + G_k(\theta)P_k^2[n] \sigma^2_{z,k}} > \sum_{k=1}^{K} \sum_{n=1}^{N} \ln(1 + \text{SNR}_k(\theta, n)),
$$

(3.37)

where $a_k^H(\theta)a_k(\theta) = G_k(\theta)\tilde{a}_k^H(\theta)\tilde{a}_k(\theta) = G_k(\theta)$, where $\tilde{a}_k(\theta)$ is a unit vector, $G_k(\theta)$ is the maximum achievable gain of the formed beam in the direction $\theta$, and $\text{SNR}_k(\theta, n) = G_k(\theta)P_k^2[n] \sigma^2_{x,k} / \sigma^2_{z,k}$ is the maximum achievable post-beamforming SNR. It is interesting to note that the right-hand side of this detector bears structural similarities to channel capacity, a concept from information theory.

### 3.4.2 Calibrated Array

When the array is fully calibrated, $a_k(\theta)$ and noise covariance $\mathbf{R}_{z,k}$ is known, or there are at least low-error estimates of them. Therefore, let $w_k(\theta) = \mathbf{R}_{z,k}^{-1}a_k(\theta)$, which is the maximum-SNR beamformer weight vector that points in the direction $\theta$ (see Section 2.3.2). Therefore,

$$
\sum_{k=1}^{K} \sum_{n=1}^{N} \frac{P_k^2[n] \sigma^2_{x,k} \mathbf{w}_k^H(\theta)x_k[n]|^2}{1 + P_k^2[n] \sigma^2_{x,k} a_k^H(\theta) \mathbf{R}_{z,k}^{-1} a_k(\theta)} > \sum_{k=1}^{K} \sum_{n=1}^{N} \ln(1 + P_k^2[n] \sigma^2_{z,k} a_k^H(\theta) \mathbf{R}_{z,k}^{-1} a_k(\theta)).
$$

(3.38)

This detector essentially requires one to maximize SNR by applying a matched filter in space, time, and frequency. Consequently, the optimal approach to transient detection in an ML sense is to search over values of $\sigma^2_{z,k} P_k^2[n]$ until the left-hand size is maximized and the right-hand side is minimized prior to performing the threshold test.

### 3.4.3 Uncalibrated Array

When $a_k(\theta)$ is unknown, but an estimate of $\mathbf{R}_{z,k}$ is available, let $\hat{x}_k[n] = \mathbf{R}_{z,k}^{-1}x_k[n]$. This is the noise-whitened version of the array voltages, and the detector in (3.31) simplifies to

$$
\sum_{k=1}^{K} \sum_{n=1}^{N} \frac{P_k^2[n] \sigma^2_{x,k} |a_k^H(\theta)\hat{x}_k[n]|^2}{1 + P_k^2[n] \sigma^2_{x,k} a_k^H(\theta) \mathbf{R}_{z,k}^{-1} a_k(\theta)} > \sum_{k=1}^{K} \sum_{n=1}^{N} \ln(1 + P_k^2[n] \sigma^2_{z,k} a_k^H(\theta) \mathbf{R}_{z,k}^{-1} a_k(\theta)).
$$

(3.39)
As with the calibrated array case, one must search over values of $\sigma_{s,k}^2 P_k^2[n]$ before performing the threshold test. However, this result also requires a search over possible array steering vectors $a_k(\theta)$, thus further complicating the computational efficiency of this detector. Such a search is impossible for an uncalibrated array, so we recommended performing array calibration before attempting any type of transient detection.

### 3.5 Transient Detection in Beam Space

The detectors derived thus far in this chapter use received element voltages, but these are often not available in a beamforming array. Instead, one may have only beamformed voltages $y_k[n]$, where

$$y_k[n] = W^H_k x_k[n],$$

(3.40)

where $W_k$ is a $J \times M$ matrix whose columns are beamformer weight vectors that result in $J$ beams. The application of weights can be thought of as a transformation into “beam space,” or the array subspace spanned by the fixed set of beamformer weights for a fixed number of $J$ beams. There are various advantages to this including the ability to filter out undesirable noise or interference through the appropriate selection of beamformer weights. If $J < M$, then beamformed voltages also reduce the overall data load to process, making downstream processing less computationally expensive. In this section, the ML detector will be derived again using beamformed voltages, where the ML criterion is defined in (3.10) with $x_n[n]$ is replaced with $y_k[n]$.

Under the two hypotheses from Section 3.4, (3.40) becomes

$$y_k[n] = \begin{cases} W^H_k z_k[n], & H_0 \text{ is true} \\ W^H_k P_k[n] s_k[n] a_k(\theta) + W^H_k z_k[n] & H_1 \text{ is true.} \end{cases}$$

(3.41)
Since the application of a beamformer is a linear transformation, the distribution of $y_k[n]$ is a zero-mean Gaussian random vector process with covariance matrix

$$R_{y,k}[n] = E\{y_k[n]y_k^H[n]\}$$

(3.42)

$$= W_k^H R_{s,k}[n] W_k.$$  
(3.43)

Thus, the covariance under each hypothesis is given by

$$R_{y,k}[n] = \begin{cases} R_{z,k}, & H_0 \text{ is true,} \\ R_{s,k}[n] + R_{z,k}, & H_1 \text{ is true,} \end{cases}$$

(3.44)

where $R_{z,k} = W_k^H R_{z,k} W_k$ and $R_{s,k}[n] = P_k^2[n] \sigma_{s,k}^2 W_k^H a_k(\theta) a_k^H(\theta) W_k$. The PDFs conditioned on each hypothesis are then given by

$$p(y_k[n]|H_0) = \frac{1}{\pi^N |R_{z,k}|} e^{-y_k^H[n]R_{z,k}^{-1}y_k[n]},$$

$$p(y_k[n]|H_1) = \frac{1}{\pi^N |R_{s,k}[n] + R_{z,k}|} e^{-y_k^H[n](R_{s,k}[n] + R_{z,k})^{-1}y_k[n]}.$$ 

Therefore,

$$\sum_{k=1}^K \sum_{n=1}^N \ln |R_{z,k}| + y_k^H[n] R_{z,k}^{-1} y_k[n] > \sum_{k=1}^K \sum_{n=1}^N \ln |R_{s,k}[n] + R_{z,k}| + y_k^H[n](R_{s,k}[n] + R_{z,k})^{-1} y_k[n].$$

In order to further simplify, we apply the Woodbury matrix identity from (3.14) and the generalized matrix determinant lemma from (3.12) to obtain

$$(R_{s,k}[n] + R_{z,k})^{-1} = R_{z,k}^{-1} - \frac{P_k^2[n] \sigma_{s,k}^2 W_k^H a_k(\theta) a_k^H(\theta) W_k R_{z,k}^{-1}}{1 + P_k^2[n] \sigma_{s,k}^2 a_k^H(\theta) W_k R_{z,k}^{-1} W_k^H a_k(\theta)},$$

(3.45)

and

$$|R_{s,k}[n] + R_{z,k}| = |1 + P_k^2[n] \sigma_{s,k}^2 a_k^H(\theta) W_k R_{z,k}^{-1} W_k^H a_k(\theta)||R_{z,k}|.$$ 

(3.46)
Hence,

\[
\sum_{k=1}^{K} \sum_{n=1}^{N} \frac{P_k^2[n] \sigma_{s,k}^2 |\mathbf{a}_k^H(\theta) \mathbf{W}_k \mathbf{z}_k^{-1} \mathbf{y}_k[n]|^2}{1 + P_k^2[n] \sigma_{s,k}^2 |\mathbf{a}_k^H(\theta) \mathbf{W}_k \mathbf{z}_k^{-1} \mathbf{W}_k^H \mathbf{a}_k(\theta)|} > \sum_{k=1}^{K} \sum_{n=1}^{N} \ln\left(1 + P_k^2[n] \sigma_{s,k}^2 |\mathbf{a}_k^H(\theta) \mathbf{W}_k \mathbf{z}_k^{-1} \mathbf{W}_k^H \mathbf{a}_k(\theta)|\right). \tag{3.47}
\]

Much like the results from Section 3.4, this detector is difficult to interpret without stating further assumptions. This detector will be evaluated in the following subsections when the noise is spatially white, the array is calibrated, and the array is uncalibrated.

### 3.5.1 Spatially White Noise

When the beam-space spatial noise is white,

\[
\mathbf{R}_{z,k} = \sigma_{z,k}^2 \mathbf{I}. \tag{3.49}
\]

This can happen when the beamformer weights are orthogonal and \( \mathbf{R}_{z,k} = \sigma_{z,k}^2 \mathbf{I} \), or when the beamformer weight matrix is square, full-rank, and is selected such that it whitens \( \mathbf{R}_{z,k} \). Note that when beam FoVs have small overlap (e.g., have widely separated main lobes), this is approximately true.

In either case, the detector from (3.48) becomes

\[
\sum_{k=1}^{K} \sum_{n=1}^{N} \frac{P_k^2[n] \sigma_{s,k}^2 |\mathbf{a}_k^H(\theta) \mathbf{W}_k \mathbf{y}_k[n]|^2}{1 + P_k^2[n] \sigma_{s,k}^2 |\mathbf{a}_k^H(\theta) \mathbf{W}_k \mathbf{W}_k^H \mathbf{a}_k(\theta)|} > \sum_{k=1}^{K} \sum_{n=1}^{N} \ln\left(1 + P_k^2[n] \sigma_{s,k}^2 |\mathbf{a}_k^H(\theta) \mathbf{W}_k \mathbf{W}_k^H \mathbf{a}_k(\theta)|\right) \tag{3.50}
\]

If the beamformer weight vectors are orthonormal, this simplifies to

\[
\sum_{k=1}^{K} \sum_{n=1}^{N} \frac{\text{SNR}_k(\theta,n) |\mathbf{\tilde{a}}_k^H(\theta) \mathbf{W}_k \mathbf{y}_k[n]|^2}{1 + \text{SNR}_k(\theta,n) |\mathbf{\tilde{a}}_k^H(\theta) \mathbf{W}_k \mathbf{W}_k^H \mathbf{\tilde{a}}_k(\theta)|} > \sum_{k=1}^{K} \sum_{n=1}^{N} \ln\left(1 + \text{SNR}_k(\theta,n) |\mathbf{\tilde{a}}_k^H(\theta) \mathbf{W}_k \mathbf{W}_k^H \mathbf{\tilde{a}}_k(\theta)|\right). \tag{3.51}
\]
which looks nearly identical to (3.37). Furthermore this can be written as

\[ \sum_{k=1}^{K} \sum_{n=1}^{N} \frac{\text{SNR}_k(\theta, n)|\mathbf{b}_k^H(\theta)\mathbf{y}_k[n]|^2}{1 + \text{SNR}_k(\theta, n)} > \sum_{k=1}^{K} \sum_{n=1}^{N} \ln(1 + \text{SNR}_k(\theta, n)), \]

(3.52)

where \( \mathbf{b}_k(\theta) = \mathbf{W}_k^H \mathbf{a}_k(\theta) \) is a beam-space steering vector. As with the element-space treatment, one must search over values of \( \mathbf{b}_k(\theta) \) (for a fixed set of weights \( \mathbf{W}_k \)) for a maximum if the array is not calibrated. However, if the rank of \( \mathbf{W}_k \) is less than \( M \), then the search can be performed over a reduced-dimensional space, making the search less computationally expensive.

### 3.5.2 Calibrated Array

Let \( \tilde{\mathbf{y}}_k[n] = \mathcal{R}_{z,k}^{-1} \mathbf{y}_k[n] \) be a noise-whitened observation vector after beamforming, and let \( \mathbf{b}_k(\theta) = \mathbf{W}_k^H \mathbf{a}_k(\theta) \) be a steering vector in beam space. Thus (3.48) becomes

\[ \sum_{k=1}^{K} \sum_{n=1}^{N} \frac{P_k^2[n] \sigma_{s,k}^2 |\mathbf{a}_k^H(\theta)\mathbf{b}_k[n]|^2}{1 + P_k^2[n] \sigma_{s,k}^2 |\mathbf{a}_k^H(\theta)\mathcal{R}_{z,k}^{-1}\mathbf{b}_k(\theta)|^2} > \sum_{k=1}^{K} \sum_{n=1}^{N} \ln(1 + P_k^2[n] \sigma_{s,k}^2 |\mathbf{a}_k^H(\theta)\mathcal{R}_{z,k}^{-1}\mathbf{b}_k(\theta)|^2). \]

(3.53)

This is a similar structure to that found in Section 3.4.3, except that the observation vector and steering vectors are now found in beam space instead of element space. As before, one must search over values of \( P_k^2[n] \sigma_{s,k}^2 \) before performing this threshold test.

### 3.5.3 Uncalibrated Array

By expanding the beam-space noise covariance, we can write

\[ \mathbf{a}_k^H(\theta) \mathbf{W}_k^H \mathbf{R}_{z,k} \mathbf{W}_k^{-1} \mathbf{y}_k[n] = \mathbf{a}_k^H(\theta) \hat{\mathbf{x}}_k[n], \]

(3.54)

where \( \hat{\mathbf{x}}_k[n] = \mathbf{W}_k(\mathbf{W}_k^H \mathbf{R}_{z,k} \mathbf{W}_k)^{-1} \mathbf{y}_k[n] \), and (3.48) becomes

\[ \sum_{k=1}^{K} \sum_{n=1}^{N} \frac{P_k^2[n] \sigma_{s,k}^2 |\mathbf{a}_k^H(\theta)\hat{\mathbf{x}}_k[n]|^2}{1 + P_k^2[n] \sigma_{s,k}^2 |\mathbf{a}_k^H(\theta)\mathbf{W}_k \mathcal{R}_{z,k}^{-1} \mathbf{W}_k^H \mathbf{a}_k(\theta)|^2} > \sum_{k=1}^{K} \sum_{n=1}^{N} \ln(1 + P_k^2[n] \sigma_{s,k}^2 |\mathbf{a}_k^H(\theta)\mathbf{W}_k \mathcal{R}_{z,k}^{-1} \mathbf{W}_k^H \mathbf{a}_k(\theta)|^2). \]

(3.55)
The $W_k(W_k^H R_{z,k} W_k)^{-1}$ term looks similar to that of a Moore-Penrose pseudoinverse, implying that $\hat{x}_k[n]$ is a vector that is in a rank-reduced subspace of the element space. An implication of this result is that one can still enjoy the spatial filtering benefits of a beamformer even if only exclusively using element voltages.

3.6 Conclusions

In this chapter, we have found ML optimal detectors and estimators for various beamforming array applications. When estimating an array steering vector, the ML estimator is identical to that used by BYU for array calibration, thus confirming that, in an unconstrained environment, the practiced approach is optimal. Transient detectors, on the other hand, traditionally attempt to apply a matched filter in time and frequency, which is an important aspect of the derived ML detectors in this chapter. However, the practiced approaches neglect to use the matched filter parameters to normalize received power, resulting in sub-optimal detection. Even when performing boxcar smoothing, which is already a sub-optimal matched filter, the parameters of the boxcar can still be used to improve detection.
CHAPTER 4. CRAMÉR RAO BOUND FOR ASTRONOMICAL TRANSIENT PARAMETERS

4.1 Introduction

The study of fast radio bursts (FRBs) represents an exciting new frontier in the radio astronomy community. To date, there is no consensus regarding the origin of these one-off, highly dispersed, broadband transients. There are, however, many theories ranging from colliding supermassive black holes [81] to merging binary white dwarf stars [80].

No one theory has yet been widely accepted due to a lack of documented observations of these mysterious pulses, which have been rare (23 documented at the time of this writing [94]) despite incident rate estimates of 2500 per day in the observable sky [2], [3]. This small number of detected incidents is arguably the result of the inherently narrow field of view (FoV) of traditional radio astronomy telescope dish horn feeds, also called single-pixel feeds since they can only image one point on the sky in an instant. Only one documented FRB has been detected using a single-beam receiver [11].

One can greatly widen the FoV of a receiver by using more antennas, such as with the Parkes Multibeam receiver [95] and the Arecibo L-Band Feed Array (ALFA) [96]. These systems replace traditional single-horn feeds in the focal plane of a parabolic dish with moderately spaced horn arrays. Each horn then creates its own beam on the sky. Multiple beams can also be formed using phased array feeds (PAFs), tightly spaced antenna arrays in the focal plane. This is achieved through beamforming, the summation of weighted individual element voltages. By beamforming with appropriate weights, the beam direction and pattern can be controlled to achieve some desirable response [68].

These array systems are incredibly expensive and must at times make sacrifices to bandwidth or temporal resolution, which can negatively affect transient detection and parameter estimator (pulsar timing algorithm) accuracy. The impact of system specifications on estimator accu-
racy and precision can be quantified using formal estimation theory. For example, one is able to determine fundamental limits on parameter estimators such as the absolute minimum achievable estimation error variance (or precision), known as the Cramér Rao Bound (CRB). An estimator which achieves this lower bound is called efficient [97], which grants one the ability to rank and compare performance of competing candidate parameter estimators. As such, a low CRB (i.e., low error variance) is preferred.

The CRB is found by taking the inverse of the Fisher information matrix (FIM), which characterizes the cross coupling of the parameters of a random variable or process. For example, the parameters of mean, median, and mode for a Gaussian random variable are identical and thus exhibit maximum mutual information. Alternatively, the parameters of mean and variance for the same random variable are completely uncoupled, resulting in zero mutual information.

Despite being an invaluable tool for estimator evaluation, the CRB has largely been unused in the practice of pulsar astronomy. We assert that, for example, an analysis of the minimum error variance can assist the astronomer or engineer in developing a new timing algorithm and illuminate which receiver parameters (e.g., bandwidth and observation frequency) make the largest impact on downstream estimation efficiency.

In this analysis, we present a formulation for the CRB of the dispersion measure (DM) parameter of astronomical transients when using either a single antenna or an array antenna. We do not present a new efficient estimation algorithm for astronomical transient parameters since the CRB is primarily an analytical tool and does not suggest a design for the minimum variance estimator. Rather, these results serve as a guide in the design of antenna receiver systems to achieve maximum potential for downstream estimation. Further, the results presented here will enable analysis of existing pulsar estimation methods to determine if there is any gap between the CRB ideal and current performance to justify continued efforts to find a lower error variance estimation algorithm.

The current standard for pulsar parameter estimation is to use the tempo2 pulsar-timing package [47], [48]. These codes precisely estimate pulse times of arrival (TOAs) using complex models that include observatory clock offsets, atmospheric propagation delays, Einstein delays (temporal shifts caused by changes in gravitational potential), and Shapiro delays (delays caused by the warping of space-time near large masses like the Sun). While the work presented here is
based on a simple dispersive model, the model and analytical framework used here in formulating the CRB can be enhanced to include these various additional parameters and provide the user with the ability to formally evaluate the efficiency of their timing algorithms.

Cordes et al. examined the effect of frequency-dependent DMs, or “Chromatic DMs,” on pulsar timing [53]. In their work, they indicate that this phenomenon results in timing residuals that can be modeled as temporally correlated noise. While this paper only considers “white” DMs, the CRB formulation presented herein can be modified to accommodate a more sophisticated model such as when the DM is frequency dependent.

Many other researchers have analyzed the optimality of DM estimators using different approaches than what is presented here including Hobbes et al., who analyzed low-frequency timing errors found in hundreds of documented pulsars and also evaluated existing models that try to account for correlated noise [49]. Coles et al. also examined several commonly used \textit{ad hoc} approaches that attempt to minimize these temporally correlated low-frequency timing residuals, showed they are sub-optimal, and introduced a least-squares fitting algorithm to whiten the residuals and more accurately estimate transient parameters [51]. Taking a more general approach, Haasteren and Levin introduced a Bayesian estimation technique that compensates for the effects of all types of temporally correlated measurement noise [52]. Each of these approaches claim superiority through mostly subjective metrics. With a full CRB analysis, however, each algorithm can be evaluated against a single objective criterion (efficiency).

Ultimately, our hope is that the techniques described here can later be used in a larger sense to model timing irregularities, frequency-dependent DMs, and temporally varying DMs. Transient timing algorithms can then be evaluated in terms of efficiency to determine if the approach can be improved.

\subsection*{4.2 Signal Model}

Before delving into the mathematical framework we will be using in our analysis, it is instructive to establish a notation. We will denote a generic scalar parameter as $\alpha$ and multiple parameters as the vector equivalent $\mathbf{\alpha}$. 
4.2.1 Single-Pixel Model

Here we present a mathematical model for a discrete-time narrowband signal received by a single-pixel antenna. The complex basebanded voltage induced at the output of an antenna at time sample \( n \), \( 0 \leq n \leq N - 1 \), in the length \( N \) short-time window (STW) \( m \), \( 1 \leq m \leq M \), by an astronomical transient point source in a single narrowband frequency channel \( k \), \( 1 \leq k \leq K \) can be modeled as

\[
v_k[mN + n] = h_k(D, m, t_0, T)s_k[mN + n]a(\theta, \phi) + z_k[mN + n]. \tag{4.1}
\]

The function \( h_k(D, m, t_0, T) \) is a deterministic but unknown time-frequency dispersed pulse shape where \( 0 \leq h_k(D, m, t_0, T) \leq 1 \) with dispersion measure \( D \) (cm\(^{-3}\) pc), pulse start time \( t_0 \) (s), and pulse width \( T \) (s). The sequence \( s_k[n] \) is a wide-sense stationary random zero-mean circular complex Gaussian process with variance \( \sigma^2_{s,k} \), and \( z_k[n] \) is additive noise modeled as a wide-sense stationary random zero-mean complex Gaussian vector process with variance \( \sigma^2_{z,k} \). Lastly, \( a(\theta, \phi) \) is the gain and phase response of the antenna to a plane wave arriving from the elevation angle \( \theta \) and azimuth angle \( \phi \). This model assumes that the STW length \( N \) is short enough that the pulse amplitude can be approximately modeled as constant over an individual STW.

This model also assumes that (1) the effects of Doppler shift due to the orbit of the Earth around the Sun are corrected, (2) any pulse period folding in the case of periodic transients like pulsars has already taken place, and (3) the true underlying DM is not frequency-dependent (i.e., it is white). This is not to say that these effects cannot be included in the model, but we have chosen to omit them for the sake of minimizing the parameter list length and simplifying this introduction to the CRB.

Assuming independence between the signal and noise amplitudes, the variance of the received voltage for STW \( m \) and frequency channel \( k \) is given by

\[
\sigma^2_{v,k,m} = E\{|v_k[mN + n]|^2\} \tag{4.2}
\]

\[
= h^2_k(D, m, t_0, T)\sigma^2_{s,k}|a_k(\theta, \phi)|^2 + \sigma^2_{z,k}, \tag{4.3}
\]
where $E\{\cdot\}$ is the expectation operator. Let $\alpha = [D, t_0, T, \sigma_{s,1}^2, \ldots, \sigma_{s,K}^2, \sigma_{s,1}^2, \ldots, \sigma_{s,K}^2] \in \mathbb{R}^{2K + 2}$ be the parameter vector containing all terms needed to fully specify the pulse across all $K$ frequency channels.

### 4.2.2 Array Antenna Model

A signal model for an antenna array (such as a PAF) consisting of $J$ constituent elements is similar to that of the single-pixel antenna and is given by

$$v_k[mN + n] = h_k(D, m, t_0, T)s_k[mN + n]a_k(\theta, \phi) + z_k[mN + n], \quad (4.4)$$

where $v_k[n]$ is a $J \times 1$ vector with entries representing the induced complex basebanded voltages in a narrowband channel $k$ for the time sample $n$, and $z_k[mN + n]$ is the array noise, which is modeled as a temporally white circularly complex Gaussian random sequence with spatial covariance matrix $R_{z,k} = E\{z_k[mN + n]z_k^H[mN + n]\}$. The matrix $R_{z,k}$ depends on, for example, individual array element receiver noise, mutual coupling between elements, and correlation caused by the spatial structure of the dish’s spillover illumination pattern (in the case of PAFs). We have also introduced the vector $a_k(\theta, \phi)$, known as a steering vector, which represents the array gain and phase response to a plane wave arriving from the 2D angle $(\theta, \phi)$.

As before, we assume independence between the signal and noise amplitudes, resulting in the spatial covariance matrix

$$R_{v,k,m} = E\{v_k[mN + n]v_k^H[mN + n]\}$$

$$= h_k^2(D, m, t_0, T)\sigma_{s,k}^2a_k(\theta, \phi)a_k^H(\theta, \phi) + R_{z,k}. \quad (4.5)$$

Under this model, there are significantly more parameters due to the spatial diversity created by the antenna array. These parameters are

$$\alpha = [D, t_0, T, \theta, \phi, \sigma_{s,1}^2, \ldots, \sigma_{s,K}^2, \text{vec}\{R_{z,0}\}^T, \ldots, \text{vec}\{R_{z,K-1}\}^T], \quad (4.7)$$
where \( \text{vec}\{\mathbf{R}_{z,k}\} \) is the vectorization operator, which stacks the columns of \( \mathbf{R}_{z,k} \) into a single large column vector.

### 4.2.3 Pulse Shape Model

There are many pulse profiles a pulsar or FRB can exhibit, but for the sake of simplicity we will assume a Gaussian pulse shape with mean \( t_0 - 2T \) and variance \( 2T \). Such a pulse has the profile

\[
h_k(D,m,t_0,T) = \exp\left\{ -\left( mT_s - (t_0 - 2T - mD(k))^2 \right) \frac{1}{4T} \right\},
\]

where \( T_s \) is the sampling period in seconds, and \( mD(k) \) is the frequency dependent delay in seconds relative to frequency index \( k = 0 \) induced by a DM of \( D \). This delay is represented by \[ \text{(4.8)} \]

\[
mD(k) = 4.1488 \times 10^3 (f_0^{-2} - f_k^{-2}) D
\]

\[
= c(f_0^{-2} - f_k^{-2}) D,
\]

where \( f_k \) is the center frequency in MHz of the \( k \)th frequency bin, and \( c \) is the “dispersion constant.”

### 4.3 Framework for Cramér Rao Bound

As stated before, the CRB is the inverse of the FIM, \( \mathbf{F} \), which, for a list of \( P \) parameters, is a \( P \times P \) matrix where the entry in the \( i \)th row and \( j \)th column is given by

\[
F_{i,j} = E \left\{ \frac{\partial}{\partial \alpha_i} \ln p(\mathbf{V}|\alpha) \frac{\partial}{\partial \alpha_j} \ln p(\mathbf{V}|\alpha) \right\},
\]

where \( p(\mathbf{V}|\alpha) \) is the probability density function (PDF) of the random matrix \( \mathbf{V} \) conditioned on the parameters \( \alpha \) and is sometimes referred to as a likelihood function, and \( \mathbf{V} \) is a matrix consisting of all time samples and frequency channels under test, which will be defined in more detail in later sections. Thus, the mutual Fisher information can be thought of as the expected coupled sensitivity of the log-likelihood function \( \ln p(\mathbf{V}|\alpha) \) to a marginal variation in the \( i \)th and \( j \)th parameters \( \alpha_i \) and \( \alpha_j \). When an undesired parameter (i.e., one which is not of interest to the user, but which
materially affects \( p(V, \alpha) \), and so must be included in the model) shares high mutual information with a desired one, the variance of the desired parameter estimate is degraded. These troublesome values are often referred to as nuisance parameters.

When \( V \) is Gaussian distributed (a valid assumption for astronomical transient signals), the entire FIM can be simplified to the compact notation,

\[
F = J^H (R_V^{-1} \otimes R_V^{-1}) J,
\]

where \( \otimes \) is the Kronecker matrix product, \( R_V = \frac{1}{N}E\{VV^H\} \) and is the covariance matrix of the received signal, \( (\cdot)^H \) is the conjugate transpose operator, and \( \overline{\cdot} \) is the conjugation operator. The matrix \( J \) is a Jacobian matrix given by

\[
J = \frac{\partial vec(R_V)}{\partial \alpha^T}.
\]

### 4.3.1 Single-Pixel Antenna

In the case of a single antenna, the matrix \( V \) is formed by stacking all time and frequency samples that are used for estimating parameters such that

\[
V = \begin{bmatrix}
    v_1[0] & \cdots & v_1[N - 1] \\
    \vdots & & \vdots \\
    v_1[(M - 1)N] & \cdots & v_1[MN - 1] \\
    \vdots & & \vdots \\
    v_k[(m - 1)N] & \cdots & v_k[mN - 1] \\
    \vdots & & \vdots \\
    v_K[(M - 1)N] & v_K[MN - 1]
\end{bmatrix},
\]

(4.14)
where each row of this matrix represents a single STW for one frequency channel. The covariance matrix for $V$ is then given by

$$
R_V = \frac{1}{N} E\{VV^H\} =
\begin{bmatrix}
\sigma^2_{v,1,1} & 0 & \cdots & 0 \\
0 & \sigma^2_{v,1,2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0
\end{bmatrix}.
$$

(4.15)

Thus, since $J$ is formed from the elements of $R_V$, the product in (4.12) can be greatly simplified to

$$
F = \tilde{J}^H R_V^{-2} \tilde{J},
$$

(4.17)

where $R_V^{-2} = R_V^{-1} R_V^{-1}$, and

$$
\tilde{J} = \left(\frac{\partial \text{diag}(R_V)}{\partial \alpha^T}\right).
$$

(4.18)

The derivation of the entries of $\tilde{J}$ for the pulse shape model in (4.8) is presented in Appendix A.1.

### 4.3.2 Array Antenna

In the case of an array antenna, the matrix $V$ becomes

$$
V =
\begin{bmatrix}
v_0[0] & \cdots & v_0[N-1] \\
\vdots & \ddots & \vdots \\
v_0[(M-1)N] & \cdots & v_0[MN-1] \\
\vdots & \ddots & \vdots \\
v_k[(m-1)N] & \cdots & v_k[mN-1] \\
\vdots & \ddots & \vdots \\
v_{K-1}[(M-1)N] & \cdots & v_{K-1}[MN-1]
\end{bmatrix},
$$

(4.19)
and the stacked covariance matrix becomes the block-diagonal matrix

\[
R_V = \begin{bmatrix}
R_{v,0,0} & 0_{J \times J} & \cdots & 0_{J \times J} \\
0_{J \times J} & R_{v,0,1} & \cdots & 0_{J \times J} \\
\vdots & \vdots & \ddots & \vdots \\
0_{J \times J} & 0_{J \times J} & \cdots & R_{v,K-1,M-1}
\end{bmatrix},
\]

where \(0_{J \times J}\) is a \(J \times J\) matrix of zeros. This block structure results in a simplified form for the FIM given by

\[
F = \sum_{k=0}^{K-1} \sum_{m=0}^{M-1} J_{k,m}^H R_{k,m} J_{k,m},
\]

where \(R_{k,m} = R_{v,k,m}^{-1} \otimes R_{v}^{-1}\), and

\[
J_{k,m} = \begin{bmatrix}
0_{(m+Mk) \times P} \\
\frac{\partial}{\partial \alpha} R_{(0,0) v,k,m} \\
\vdots \\
\frac{\partial}{\partial \alpha} R_{(J-1,J-1) v,k,m} \\
0_{((KM-1)J^2-(m+Mk)J) \times P}
\end{bmatrix},
\]

where \(r_{v,k,m}^{(i,j)}\) is the \((i,j)\)th entry of the matrix \(R_{v,k,m}\). This quantity is given by

\[
r_{v,k,m}^{(i,j)} = E\{v_k^{(i)}[mN+n]v_k^{*(j)}[mN+n]\}
\]

\[
= h_k^2(D,m,t_0,T) \sigma_{\delta,k}^2 a_k^{(i)}(\theta,\phi) a_k^{*(j)}(\theta,\phi) + r_{z,k}^{(i,j)},
\]

where \(v_k^{(i)}[n]\) is the \(i\)th entry of \(v_k[n]\), \(a_k^{(j)}(\theta,\phi)\) is the \(j\)th entry of \(a_k(\theta,\phi)\), and \(a_k^{*(j)}(\theta,\phi)\) is the complex conjugate of \(a_k^{(i)}(\theta,\phi)\). The derivation of the entries of \(J_{k,m}\) for the pulse shape model in (4.8) are presented in Appendix A.2.
4.4 Methods

We use MATLAB scripts to populate the entries of the FIM according to the above formulations for a single antenna and for an array antenna. In these formulations, we compute the FIM as a function of the true underlying values of various parameters including DM, pulse start time, observation frequency, total bandwidth, frequency resolution, pulse width, temporal resolution, total time span, and arrival angle. We then invert the resulting FIM to obtain the CRB and extract the entry corresponding to the DM parameter.

Additionally, we explore the scenario in which only a fraction of the total bandwidth could be acquired. This fractional bandwidth, however, need not be contiguous when dealing with frequency channelized data, resulting in an enormous number of potential channel subsets. Consequently, we select a small sampling of candidate channel subsets and examine the effect on the CRB for each. This scenario could arise in array systems like the FLAG PAF where I/O data rate and memory storage capacity limitations (e.g., where operating in comensal mode with both a correlator and real-time beamformer) could force a selection of some few subbands within the instrument’s full bandwidth. This analysis will provide guidance for how to distribute these subbands across the full bandwidth without significant increases in estimation error variance.

Lastly, in all of these results, we simulate a large parabolic reflector similar to the Green Bank Telescope (GBT) with the feed at prime focus. Specifically, we use a dish diameter of 100 meters, a focal-length-to-diameter ratio \((f/D)\) of 0.7, and a non-offset reflector. For the array cases, we use a detailed numerical electromagnetic solver that generates array steering vectors given array element fields and impedances and reflector optic parameters such as dish diameter and \(f/D\). This numerical code incorporates the effects of inter-element mutual coupling, LNA noise and its recoupling, antenna loss, and spillover noise [66].

4.5 Results

The results presented in this section are the CRB for DM as a function of the following parameters:
1. Processing window size
2. Pulse arrival time
3. Pulse DM
4. Frequency
5. Total contiguous bandwidth
6. Fractional non-contiguous bandwidth
7. Frequency channel resolution
8. Pulse width
9. Temporal resolution
10. Signal-to-noise ratio (SNR)
11. Elevation angle of arrival

For the sake of brevity, the CRBs for items 1-10 are those of the single-pixel case described in Sections 4.2.1 and 4.3.1. The CRB for item 11, however, will be that of the array antenna model from Sections 4.2.2 and 4.3.2, since the primary benefits of an array antenna is an increase in angular FoV.

4.5.1 Window Size

Figure 4.1 depicts the CRB when the processing window size (i.e., $MT_s$ is varied). In this figure and those to follow, we have also superimposed corresponding time-frequency plots of the pulse under test to help illustrate what is being tested. Here we see an enormous slope of -250 dB/decade in the CRB when the window size is less than the pulse width plus the pulse start time (i.e., $t_0 + T$, marked by the vertical dashed line), which is where the pulse is partially outside the processing window. This indicates that a tremendous amount of information related to DM is lost in that region. For sufficiently large window sizes (i.e., larger than the pulse width plus the pulse start time), the CRB is minimized.
Figure 4.1: Effect of total window size on CRB. The vertical line indicates when the pulse was contained in the window (i.e., \( t_0 + T \)).

These behaviors indicate that DM estimates are guaranteed to have very high variance when the pulse is not fully contained in the processing window, and that there is no additional reduction in estimator variance with processing windows that are larger than the pulse width. While these conclusions may seem obvious without a formal CRB analysis, the results serve as an example of the utility of the CRB in providing insight into how system parameters affect downstream data reduction, analysis, and parameter estimation.

4.5.2 Pulse Start Time

The effect of the pulse start time \( t_0 \) on the CRB is depicted in Figure 4.2. The vertical dashed lines in this plot indicate where 90% of the pulse’s power is present in the window, or,
in other words, they roughly indicate when a significant portion of pulse is outside of the processing window. As with the results from Section 4.5.1, the CRB undergoes a dramatic increase (-250 dB/decade for early arrival times and 250 dB/decade for late arrival times) when the pulse is only partially contained in the processing window.

### 4.5.3 Dispersion Measure

The CRB for a sweep of DM values is shown in Figure 4.3. For each tested DM, the processing window size and pulse arrival time is selected so as to guarantee that the pulse is fully contained. An increase in the true underlying dispersion measure does not affect the minimum variance of its estimate. Further, the minimum variance as a percentage of the true DM becomes
smaller as the actual DM increases. These observations indicate that timing algorithms need not make special considerations for high DMs when they meet the CRB.

**4.5.4 Frequency**

Figure 4.4 shows the CRB resulting from a frequency sweep, where the horizontal axis denotes the lowest observed frequency and the total bandwidth observed (500 MHz) is constant across all cases. Here we observe that as the observation frequency increases, the minimum error variance for a dispersion measure estimate greatly increases at a slope of approximately 70 dB/decade. This is an understood phenomenon stemming from the standard dispersion model shown in (4.9), where the parabolic slope of a dispersed pulse makes the dispersive effect less prominent at higher frequencies [1].
4.5.5 Contiguous Bandwidth

The effect of total contiguous processing bandwidth on the CRB is shown in Figure 4.5. We ensure that the frequency channel resolution is held constant for each tested bandwidth (i.e., more bandwidth results in more frequency channels) so as to have a single independent variable. This steady change of -30 dB/decade in estimator minimum error variance as bandwidth is increased is intuitive and indicates that increasing overall processing bandwidth is vital to minimizing error variance.

4.5.6 Fractional Non-Contiguous Bandwidth

As stated before, it may only be possible to process a fraction of the total available bandwidth due to computational limits, I/O bandwidth limitations, or insufficient disk write speeds.
When working with frequency channelized data, one can use an arbitrary subset of the bandwidth, which raises the question of which subset of frequency channels will minimize the CRB. We tested five candidate subsets consisting of 1/10th of the total bandwidth with the arrangements shown in Figure 4.6. These channel subset arrangements can be described as

1. Spaced equally across band
2. Clustered equally at both ends of the band
3. Clustered into 10 equally spaced chunks across band
4. Clustered at lower end of band
5. Clustered at upper end of band
We also include the CRB for the full, unsegmented bandwidth, and the results are shown in Figure 4.7. As expected, the full bandwidth results in maximum information and minimum DM estimator variance. When only able to process a fraction of the bandwidth, however, it is not ideal to keep the channels contiguous as with masks four and five, which experience an increase of approximately 28.7 and 31.3 dB in the CRB. In our tested cases, it is preferred to have even clusters of contiguous channels placed at the edges of the band, which suffers only a 5.7 dB penalty in minimum error variance compared to the full bandwidth case. This aligns well with the current practice to use widely separated available frequencies for DM estimation [47], [50], [98], [99].
4.5.7 Frequency Resolution

The curve in Figure 4.8 shows the effect of frequency resolution on the CRB for DM. Note that for each tested frequency resolution, the total processed bandwidth is held constant. On a gross scale, the CRB is unaffected by the frequency resolution, whereas on a fine scale there is some marginal increase in the CRB as the resolution becomes largely coarse. Since this increase is minimal, these results imply that coarser channels are sufficient for DM estimation.

4.5.8 Pulse Width

Figure 4.9 shows the CRB for DM as the pulse width varies. Here we see that as the pulse width gets broader, the minimum DM estimator variance increases. This is an intuitive behavior: as the pulse widens, an ambiguity between pulse start time and dispersion measure emerges that is
increasingly sensitive to noise. A similar ambiguity arises between range and Doppler after pulse compression of a frequency-modulated (FM) radar chirp. This phenomenon is well understood by studying the dispersion measure transform [22], and is typically compensated for though boxcar fitting to minimize the pulse width [100]–[102].

4.5.9 Temporal Resolution

The minimum DM estimator variance as temporal resolution is varied is shown in Figure 4.10. We have forced the temporal resolution to be at least 1/10th of the pulse width to guarantee sufficient sampling coverage. From this we see that the temporal resolution has little impact on the CRB when the pulse width is at least 10 times as large as the sample period.
This is not the case, however, when the temporal resolution is coarser relative to the pulse width, as shown in Figure 4.11, which depicts the CRB as a function of both pulse width and sampling period. Consequently, it is desirable to have as many samples as possible per pulse width to minimize estimator variance across all classes of pulse widths.

From a practical standpoint, finer temporal resolution may not be achievable due to computational limitations, and so the maximum sample period for a given minimum pulse width would be invaluable. Figure 4.12 depicts the sampling intervals that yield the minimum CRB as a function of pulse width. In these data, we see a parabolic relationship between the pulse width and CRB-minimizing sample period that, after a regression analysis, approximately fits

\[ T_s \approx \sqrt{T}. \]  \hspace{1cm} (4.25)
These curves can help system engineers make informed performance tradeoffs in new system designs to best match temporal resolution with the target science (i.e., desired pulse widths).

4.5.10 Signal-to-Noise Ratio

Figure 4.13 depicts the effect of SNR on the minimum error variance of DM estimators. When SNR is less than unity, we see a large negative slope of approximately -40 dB/decade, indicating that decreasing noise or amplified signal strength through the use of larger apertures, pulse folding, or other signal processing methods, will dramatically reduce the minimum estimator variance for weak pulses. However, when SNR becomes larger than unity, the CRB curve flattens, indicating no additional reduction in estimation error variance.
This lack of additional decrease in the CRB for high SNR is surprising since lower noise power typically results in lower estimator variance. However, since the parameter of interest is the parabolic slope of the pulse and not a variance-related parameter such as signal power or pulse amplitude, SNR increases above unity will not dramatically improve DM estimates.

4.5.11 Angle of Arrival

The effect of the angle of arrival (AOA) on the CRB is depicted in Figure 4.14, and for reference, we have also shown the maximum achievable SNR for the given scenario by applying the maximum-SNR beamformer weights to the array data (see Section 2.3.2 for more details about beamformer weights).
To minimize computation time, we present the CRB as a function of only elevation angle. Here we see that when using an array antenna, which inherently comes with a broader FoV, the CRB is lower for AOAs that are outside the FoV of a traditional single-pixel feed (approximately 0.17 degrees for 0.7 aperture efficiency on 100 meter dish). Surprisingly, the CRB for an array antenna indicates that there is relevant DM information with sufficient SNR many beamwidths away from the main dish pointing angle.

The 3-dB FoV of the feed, or the angular region in which SNR does not descend more than 3 dB below peak, is marked with dashed vertical lines. It is in this region that one typically forms beams. We have also plotted solid vertical lines to denote where the CRB increases 3 dB from the minimum. Here we see that the SNR definition of the FoV is narrower than that of the CRB, indicating that if a sufficiently strong pulse is fortuitously detected outside of the FoV, one can
still estimate DM with low minimum variance. This is particularly encouraging for FRB detection where one never knows where to look. A PAF can be calibrated to form beams in these fringe regions so DOA information can be recovered.

Note that the SNR is substantial ($\approx 30$ dB) in these results, which is not always realistic with single pulses. A more realistic scenario is depicted in Figure 4.15, where the SNR is approximately $0$ dB. Here we see that the $3$ dB width of the CRB curve has narrowed and almost matches that of the SNR curve.

### 4.6 Conclusions

In this chapter, we have explored the CRB for the DM of single-pulse astronomical transients. From our analysis, we have learned which parameters are most strongly tied to DM includ-
ing (1) processing window size, (2) pulse start time, (3) frequency, (4) bandwidth, and (5) SNR. Note that all of these parameters are directly set or influenced by the design of the antenna receiver system and data reduction codes. Using these results, we can actually rank which system parameters are the most critical ones to optimize when designing a timing instrument. We summarize this ranking in Table 4.1. Ultimately, the most important requirements for a timing instrument from a DM estimation perspective are, in order of priority, the ability to (1) resolve the pulse in time, (2) select observation frequencies that are as low as possible, (3) minimize receiver noise and increase antenna gain to maximize SNR, and (4) increase the acquired analog bandwidth.

Furthermore, we have explored the CRB for DM when using an array antenna and found that the 3 dB FoV of the CRB can be much wider than the traditional definition of the 3 dB FoV when the pulse is sufficiently strong. Consequently, it is advisable to form beams just outside the traditional FoV in the hopes of estimating parameters if a strong pulse happens to be observed.
Figure 4.15: Effect of arrival angle on CRB with lower S/N.

Table 4.1: Major Effects of Parameters on CRB for DM

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Largest Slope</th>
<th>Rank</th>
<th>Design Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse start time</td>
<td>250 dB/decade</td>
<td>1</td>
<td>Analyze data where pulse is present</td>
</tr>
<tr>
<td>Window size</td>
<td>250 dB/decade</td>
<td>1</td>
<td>Large enough to contain pulse</td>
</tr>
<tr>
<td>Sampling period</td>
<td>250 dB/decade</td>
<td>1</td>
<td>Fine enough to sample pulse</td>
</tr>
<tr>
<td>Frequency</td>
<td>70 dB/decade</td>
<td>2</td>
<td>Lower frequency</td>
</tr>
<tr>
<td>SNR</td>
<td>40 dB/decade</td>
<td>3</td>
<td>Minimize receiver noise and boost gain</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>30 dB/decade</td>
<td>4</td>
<td>Larger bandwidth</td>
</tr>
<tr>
<td>Pulse width</td>
<td>5 dB/decade</td>
<td>5</td>
<td>Use boxcar fitting</td>
</tr>
<tr>
<td>DM</td>
<td>0 dB/decade</td>
<td>6</td>
<td>N/A</td>
</tr>
<tr>
<td>Frequency resolution</td>
<td>0 dB/decade</td>
<td>6</td>
<td>Moderately fine channels</td>
</tr>
</tbody>
</table>
Lastly, this work can be further extended to accommodate more realistic scenarios including more exotic pulse shape models, chromatic DMs, and phase jitter. Such scenarios were omitted from this treatment for the sake of focusing on the value and development of the CRB.
5.1 Introduction

The Focal L-Band Array for the Green Bank Telescope (FLAG) is a 150-MHz 38-element phased array feed (PAF) analog receiver and digital processor for the purposes of conducting neutral hydrogen (HI) surveys and astronomical radio transient searches. It is expected to be instrumental in discovering more than 50 new pulsars within the inner galactic plane [103] and studying diffuse HI (large clouds of HI with no obvious boundaries) around galaxies [104].

The entire PAF instrument comes in two parts: the analog receiver (antennas, amplification, downconversion, sampling, and signal transport) and the digital processor (channelization, beamforming, and correlation). A block diagram showing both the analog receiver and digital processor is shown in Figure 5.1. The PAF consists of 19 dual-polarization flared dipoles (see Figure 5.2) connected to 40 cryogenically cooled low-noise amplifiers (LNAs), two of which are unconnected. The 40 amplified analog bandpass signals are mixed down to baseband into inphase (I) and quadrature (Q) components, each of which are sampled at 155.52 MHz resulting in a total bandwidth of 155.52 MHz. The digitized I/Q signals are then serialized and sent over 40 optical fibers to five eight-port optical receiver cards connected to five field-programmable gate array (FPGA) boards. These FPGA boards were designed and manufactured by the Collaboration for Astronomy Signal Processing and Electronics Research (CASPER) group and are named Reconfigurable Open Architecture Computing Hardware (ROACH) boards [105]–[107].

The ROACH boards channelize the approximately 150-MHz sampled bandwidth into 512 frequency channels, each with a bandwidth of approximately 303 kHz. Six frequency channels from each of the upper and lower band edges of the 150 MHz passband are discarded, resulting in 500 remaining frequency channels. These are then packetized into 20 user-datagram protocol (UDP) packets each containing 25 frequency samples corresponding to eight antennas across 20
Figure 5.1: High-level analog receiver and digital processor block diagram.

Figure 5.2: A single GBT-2 dipole.
time samples. These packets are then streamed over 10-GbE/40-GbE breakout cables into a 12-port 40-GbE network switch, which redirects packets into five high-performance computers (HPCs) such that each HPC receives 100 frequency samples corresponding to all 40 antennas. The HPCs then take these 100 frequency samples and divide them evenly between four pipeline processes. Each pair of pipelines utilizes two graphics processing units (GPUs) to implement a real-time beamformer, filterbank channelizer, and correlator.

In this chapter, we explain the processing chain in the HPCs and show performance results from the various commissioning campaigns for the full system. Section 5.2 describes the different operational modes FLAG supports. Section 5.3 goes over HPC specifications and those of its peripherals (such as GPU card models and specs). Section 5.4 describes the pseudo real-time system known as HASHPIPE used to pipeline the process and manage shared memory. Section 5.5 briefly introduces the GPU-based real-time correlator library xGPU, and the total-power GPU library. Section 5.6 then present results from the first commissioning test of the PAF with a reduced-bandwidth version of the back end in July 2016. Section 5.7 outlines the results from the May 2017 commissioning, where many of the hardware issues from July 2016 had been resolved. Results from the final August 2017 commissioning are then provided in Section 5.8, followed by some brief conclusions and future work in Section 5.9.

5.2 Operational Modes

There are two primary modes of operation: coarse-channel mode and fine-channel mode. The coarse-channel mode produces spatial covariance matrices using 303.75 kHz channels for radio transient searches and PAF beamformer weight calibration. The fine-channel mode computes correlation matrices for 160 9.5-kHz channels for HI surveys. Both modes are specified to incorporate a concurrently running real-time beamformer that forms seven dual-polarized beams and calculate time-average power every 0.1 ms, creating a beamformed spectrometer. Figure 5.3 shows a full block diagram of these modes.
5.2.1 Coarse-Channel Mode

In this mode, each GPU processes 25 coarse channels for all 40 antenna elements, totaling approximately 7.5 MHz of bandwidth. For PAF calibration, correlation matrices are computed for all 25 channels and accumulated for a user-specified integration length, where 303 samples would correspond to 1 ms. For transient searches, matrices are computed for only five channels for 0.13 ms, or 40 samples. Only five channels (a bandwidth of 1.51 MHz) are processed since the required I/O bandwidth demand for covariance dumps every 0.13 ms exceeds the specified limits of the GPU cards.

A real-time beamformer runs in parallel to the correlator, forming seven beams on all 25 coarse channels and dumping time-averaged power every 0.13 ms. All 25 channels can be processed here, unlike with the correlator, since only beamformed time-averaged powers are saved ($\mathcal{O}(N)$) instead of cross-correlations across all elements ($\mathcal{O}(N^2)$).

Figure 5.3: Block diagram describing the various operational modes.
5.2.2 Fine-Channel Mode

This mode has the same structure as the coarse-channel mode except that it includes a fine polyphased filterbank (PFB) module for further channelization prior to the correlator. The PFB channelizes five coarse frequency channels (303.75 kHz each, totaling 1.51 MHz) into 160 finer channels (9.5 kHz each, totaling 1.51 MHz). These are then correlated, and the resulting matrices are dumped every 0.5 seconds (at minimum) or approximately 4,734 samples. All 160 fine-channel correlation matrices are loaded off the GPU for FITS file formatting. As with the coarse-channel mode, a real-time beamformer runs in parallel.

5.3 HPC Specifications

Each HPC is a Mercury GPU408 4U GPU Server from Advanced HPC (part number AH-GPU408-SB14). A photo of one of these HPCs is shown in Figure 5.4, and specifications are shown in Table 5.1. The machine comes equipped with two Intel Xeon six-core processors organized onto two independent PCI-E busses. The PCI-E slots are populated with two GPU cards, two dual-port 10-GbE SFP+ network interface controller (NIC) cards, and an Infiniband NIC card, with part designations summarized in Table 5.2.
Table 5.1: HPC Specifications

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processors</td>
<td>2x Intel Xeon E5-2630 v2 2.60 GHz six-core 80 W processors</td>
</tr>
<tr>
<td>Memory</td>
<td>32 GB DDR3 ECC</td>
</tr>
<tr>
<td>Hard Drives</td>
<td>2x 500 GB SATA 7200 RPM disks in RAID 1 configuration</td>
</tr>
<tr>
<td>Drive Bays</td>
<td>8x hot-swappable 3.5 inch drive bays (unpopulated)</td>
</tr>
<tr>
<td>PCI-E Slots</td>
<td>4x PCI-E 3.0 x16 slots (double width, two per processor)</td>
</tr>
<tr>
<td></td>
<td>2x PCI-E 3.0 x8 slots (one per processor)</td>
</tr>
<tr>
<td></td>
<td>1x PCI-E 2.0 x4 slot</td>
</tr>
<tr>
<td>Network</td>
<td>Integrated Intel i350 dual port GbE LAN</td>
</tr>
<tr>
<td>Power Supply</td>
<td>1620 W platinum level efficiency redundant power supply</td>
</tr>
<tr>
<td>Height</td>
<td>4U</td>
</tr>
<tr>
<td>Rack Mountable</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 5.2: PCI-E Slot Population (out-of-date)

<table>
<thead>
<tr>
<th>Slot Designation</th>
<th>Populated With</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU 1 Slot 2*</td>
<td>MCX312A-XCBT Dual-Port 10 Gigabit Ethernet Adapter</td>
</tr>
<tr>
<td>CPU 1 Slot 4*</td>
<td>EVGA GeForce GTX 780/980 Ti Graphics Card</td>
</tr>
<tr>
<td>CPU 2 Slot 6*</td>
<td>MCX312A-XCBT Dual-Port 10 Gigabit Ethernet Adapter</td>
</tr>
<tr>
<td>CPU 2 Slot 8*</td>
<td>EVGA GeForce GTX 780/980 Ti Graphics Card</td>
</tr>
<tr>
<td>CPU 2 Slot 9†</td>
<td>MCX353A-QCBT InfiniBand Adapter Card</td>
</tr>
<tr>
<td>CPU 1 Slot 10+</td>
<td>Unpopulated</td>
</tr>
<tr>
<td>CPU 2 Slot 11†</td>
<td>Unpopulated</td>
</tr>
</tbody>
</table>

* PCI-E 3.0 x16 slot.  † PCI-E 3.0 x8 slot.  † PCI-E 2.0 x4 slot.

5.3.1 Graphics Cards

There are two flavors of graphics cards used in the FLAG back end: the NVIDIA GeForce GTX 780 Ti and NVIDIA GeForce GTX 980 Ti. Two of the GPUs are the 780 Ti model since they were purchased just prior to NVIDIA’s abrupt removal of them from the market. Pertinent specifications for both cards are summarized in Tables 5.3 and 5.4. The differences between the two cards in all critical performance specifications are minimal.
Table 5.3: GeForce GTX 780 Ti Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td># CUDA Cores</td>
<td>2880</td>
</tr>
<tr>
<td>Base Clock Speed</td>
<td>875 MHz</td>
</tr>
<tr>
<td>Texture Fill Rate</td>
<td>210 GigaTexels/sec</td>
</tr>
<tr>
<td>Memory Clock Speed</td>
<td>7.0 Gbps</td>
</tr>
<tr>
<td>Memory</td>
<td>3 GB</td>
</tr>
<tr>
<td>Memory Interface Width</td>
<td>384 bits</td>
</tr>
<tr>
<td>Memory Bandwidth</td>
<td>336 GB/sec</td>
</tr>
<tr>
<td>Minimum Power Requirement</td>
<td>600 W</td>
</tr>
</tbody>
</table>

Table 5.4: GeForce GTX 980 Ti Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td># CUDA Cores</td>
<td>2816</td>
</tr>
<tr>
<td>Base Clock Speed</td>
<td>1000 MHz</td>
</tr>
<tr>
<td>Texture Fill Rate</td>
<td>176 GigaTexels/sec</td>
</tr>
<tr>
<td>Memory Clock Speed</td>
<td>7.0 Gbps</td>
</tr>
<tr>
<td>Memory</td>
<td>6 GB</td>
</tr>
<tr>
<td>Memory Interface Width</td>
<td>384 bits</td>
</tr>
<tr>
<td>Memory Bandwidth</td>
<td>336.5 GB/sec</td>
</tr>
<tr>
<td>Minimum Power Requirement</td>
<td>600 W</td>
</tr>
</tbody>
</table>

5.3.2 Network Cards

Each HPC is equipped with three network cards: two dual-port SFP+ 10-GbE cards for receiving packets from the ROACH boards and an InfiniBand card for output data product transfer to a high-speed disk array, known as the Lustre file system (see Figure 5.3).

5.4 HASHPIPE

HASHPIPE is a thread and shared-memory management software package that specializes in pipeline processing by splitting consecutive tasks into separate threads with semaphore-controlled shared memory buffers in between each thread. A generic HASHPIPE program block diagram is shown in Figure 5.5. Each task in a process is given its own parallel thread with input and output shared memory buffers.
Each inter-thread buffer is circular and consists of a number of blocks of data, where each block is sized according to the needs of the next thread. Each block has an associated semaphore that is initialized to zero (unavailable/free) and is set to one (available/filled) when the block contains valid data that is not being accessed by any other thread. If the block’s semaphore is set to zero, any thread that requests that block’s data will hang until the semaphore becomes available.

5.5 GPU Libraries

Each major function of the HPC pipeline process is performed in a GPU. These functions include an array correlator, total power meter, real-time seven-beam beamformer, and fine PFB channelization. This section will focus more on the correlator codes.
The real-time correlator is based on the open-source GPU library xGPU hosted on GitHub [108]. It is a highly optimized code that parallelizes the correlator process by computing several two-by-two correlations and accumulating. The output data structure is a block lower-triangular matrix with two-by-two block matrices in row-major order, and the block entries are ordered similarly. Figure 5.6 depicts the correlator output format.

There are three types of correlators that this library supports: (1) a coarse-channel correlator with 8-bit input samples, (2) a fine-channel correlator with 32-bit floating-point input samples, and (3) a coarse-channel rapid-dump, reduced-bandwidth correlator with 8-bit input samples. To accommodate these various modes, multiple versions of the library were compiled.

### 5.6 July 2016 Commissioning Results

The first commissioning of the fully assembled FLAG PAF, shown in Figure 5.7, was performed in July 2016. This included tests by the National Radio Astronomy Observatory (NRAO) team in charge of the front-end electronics and FPGA firmware to verify that portion of the system. The BYU back end was also deployed in a reduced-functionality state: only 3/10ths of the total
Figure 5.7: A picture of the array feed in the outdoor test facility prior to installation on the dish.

analog bandwidth (approximately 45 MHz of 150 MHz) was acquired, and only the coarse-channel real-time correlator was used.

Unfortunately, there were several issues with the front-end electronics including (1) oscillations in the primary amplifier voltage source, (2) optical fiber deserialization bit alignment (known as “bit lock”) errors that would occur when reprogramming the FPGAs or observing a strong source, and (3) a bug in the FPGA firmware that prevented the acquisition of subsequent scans without reprogramming the FPGAs.

Due to these shortcomings and the need for the front-end electronics team to run their own tests, results from the BYU back end system were limited. However, the coarse-channel correlator was able to acquire data for a long scan while rastering the dish around a strong point-source (also
known as a calibrator) 3C295. From this, the PAF was calibrated, beamformer weight vectors were formulated, and formed beam sensitivities were computed.

A sensitivity map using the Y-polarized elements from one of these calibration sessions is shown in Figure 5.8. The formed beam sensitivity (m^2/K) was calculated on a per-frequency channel basis for each 2D angle of arrival $\theta$ according to

$$S_k(\theta) = \frac{\eta A_p}{T_{\text{sys}}}$$

$$= \frac{2k_B}{10^{-26} F_{s,k}} \text{SNR}_k(\theta),$$

(5.1)

where $A_p$ is the physical aperture area in m^2, $\eta$ is the antenna efficiency (which is the product of the aperture efficiency and radiation efficiency), $T_{\text{sys}}$ is the system noise temperature in Kelvin (K). The constant $k_B = 1.38 \times 10^{-23}\text{ m}^2\text{kg s}^{-2}\text{K}^{-1}$ is the Boltzmann constant, $F_{s,k}$ is the flux density of the calibrator in Jy, or $10^{-26}\text{ W/m}^2$ at the center frequency of the channel $k$, and

$$\text{SNR}_k(\theta) = \frac{w_k^H(\theta)(\hat{\mathbf{R}}_{\text{on,k}}(\theta) - \hat{\mathbf{R}}_{\text{off,k}})w_k(\theta)}{w_k^H(\theta)\hat{\mathbf{R}}_{\text{off,k}}w_k(\theta)},$$

(5.2)

where $\hat{\mathbf{R}}_{\text{on}}(\theta)$ is the estimated covariance matrix when steered at an angle $\theta$ onto a strong point source, and $\hat{\mathbf{R}}_{\text{off}}$ is the sample correlation matrix resulting from looking at cold sky. The beamformer weights $w_k^H(\theta)$ are the maximum-SNR weights described in Section 2.3.2 for a beam pointing in the direction $\theta$. The flux density of the source was computed using the models provided by Perley and Butler [109].

To better understand the map in Figure 5.8, we provide an element layout diagram shown in Figure 5.9. All elements that experienced either “bit lock” errors or poor amplifier gain were omitted from the calculation of the weights and sensitivity. The X-polarized elements experienced a large number of bit-alignment errors, so the results shown in this and the following plots are only for the functioning Y-polarized elements (i.e., all but elements 12 and 14).

When there are no failed LNAs or bit-alignment errors, the sensitivity map is typically uniform within the array’s field of view (FoV). However, in this observation campaign, there were many LNAs that had spurious oscillations, and there were many unresolvable bit-lock errors, rendering many elements unusable for beamforming. Consequently this results in the non-uniform
Figure 5.8: The sensitivity map for a calibration grid on 3C295.

A blotchy map shown in Figure 5.8 and a poor peak sensitivity of 92.94 K/m² with a corresponding \( T_{\text{sys}}/\eta \) of 84.5 K. Some of the features of this map are predictable including the notches at the bottom and lower-left corner of the map where the failed elements 14Y and 12Y are respectively located. There only appear to be a few elements that are enhancing the beamformed sensitivity, namely elements 2Y, 4Y, 16Y, 19Y. Judging from the notch in its array position, 10Y appears to have also failed. Most concerning of all is the low sensitivity near the first element, which has the largest impact on overall sensitivity.

Using this same calibration data, we mapped the element patterns for the Y-polarized antennas (less 12Y and 14Y due to failures). The \( j \)th element pattern map was computed according to

\[
I_{k,\text{element}}^{(j)}(\theta) = |e_j^H \hat{a}_k(\theta)|^2, \quad (5.4)
\]
where \( \mathbf{e}_j \) is an \( M \times 1 \) vector of zeros with the \( j \)th entry equal to one, and \( \hat{\mathbf{a}}_k(\theta) \) is the estimated array steering vector for a plane wave arriving from angle \( \theta \). The array steering vector was estimated as described in Sections 2.4 and 3.3, and the resulting maps are shown in Figure 5.10.

We also mapped out formed beam patterns, with the \( i \)th formed beam pattern map computed according to

\[
I_{k,\text{beam}}^{(i)}(\theta) = |\mathbf{w}(\theta_i)^H \hat{\mathbf{a}}_k(\theta)|^2, \tag{5.5}
\]

where \( \mathbf{w}_i^H \) is the maximum-SNR beamformer weight vector for a beam pointed in the direction \( \theta_i \). A total of seven beam patterns were measured, where they were steered in a hexagonal grid as shown in Figure 5.11. The resulting patterns for seven selected locations are shown in Figure 5.12.

Unfortunately, all of the maps exhibit a jagged structure caused by a combination of two system problems: (1) the inability to perform subsequent scans without reprogramming the FPGA,
Figure 5.10: Element pattern maps from a calibration grid on 3C295.
Figure 5.11: The formed beam layout on the sky.

Figure 5.12: Formed beam pattern maps from a calibration grid on 3C295.
resulting in the need for a continuous serpentine scan, and (2) an unknown time delay between the recorded antenna positions and the acquired data. This delay was corrected heuristically until the patterns appeared to coalesce.

Ultimately, this commissioning run was considered to be only moderately successful. The reduced-functionality real-time signal processing back end independently performed very well and was able to compute spatial correlation matrices with a dump time as low as 13 ms. There were more operational modes that were not tested during this period, including the real-time seven-beam beamformer and fine-channel correlator modes.

5.7 May 2017 Commissioning Results

After a significant troubleshooting period, the NRAO team in charge of the array front end electronics was able to identify the causes of the LNA oscillations and bit-alignment errors. These fixes were verified by the NRAO team via a separate commissioning run. The array was then again installed on the GBT on May 24, 2017, for another commissioning campaign using the BYU signal processing back end, now with the full 150 MHz of bandwidth and the previously missing modes: the real-time beamformer and fine channel correlator.

The issue with the FPGA firmware that prevented subsequent scans without a reprogram cycle was resolved, and, consequently, the calibration maps were easier to manage. Also, the bit-alignment process was somewhat more streamlined, which reduced overhead and allowed us to acquire much more data. Unfortunately, one of the five HPCs experienced a hardware fault with a power distribution board, causing the affected HPC to reboot when running intensive GPU processes. Consequently, one-fifth of the bandwidth was not acquired, resulting in frequency gaps in the results that follow.

5.7.1 Outdoor Test Facility Experiments

Prior to the actual installation of the PAF on the GBT, the antennas were placed in a small test range named the Outdoor Test Facility (OTF). The room houses a feed mount with actuators that allow the feed to be raised, lowered, rotated, and tilted. Additionally, there is a retractable portion of the roof that is lined with RF absorber that can act as a hot load. When the roof is
retracted and the feed raised up, the array sees the cold darkness of space, which serves as a cold load. Several hot and cold-loaded data sets were acquired to compute “Y factors,” the ratio of the received power when observing the hot load to that when observing the cold load. This enables us to calculate the per-element system noise temperature \[ T_{sys} \], or

\[
T_{sys} = \frac{T_{hot} - Y T_{sky}}{Y - 1},
\]

where \( T_{hot} \) is the brightness temperature of the hot load (typically assumed to be 290 K), and \( T_{sky} \) is the brightness temperature of the cold sky (assumed to be 7.5 K).

The per-element averaged \( T_{sys} \) measurements in the OTF are shown in Figure 5.13. The blanked frequencies are the result of both the missing HPC referenced earlier and our blanking of apparently RFI-corrupted channels. In the nearly flat regions of this family of curves, the \( T_{sys} \) values range from approximately 13 and 20 K. Some elements did have much higher noise temperatures due to having a failed LNA or being disconnected altogether. There are also several spurious spikes near the center of the plot, which are likely caused by one of many effects: (1) DC bias and local oscillator leakage, (2) upper and lower sideband leakage in the overlap, and (3) impinging narrowband RFI.

In the fringes of the curves (i.e., the lowest and highest frequencies) in which the analog filter roll-off occurs, \( T_{sys} \) should increase due to a decrease in gain and thus \( Y \). However, here \( T_{sys} \) appears to decrease. It was determined that this behavior was caused by bit underflow when the data in the ROACH are reduced to 8-bit/8-bit real and imaginary values prior to packetization. This is corrected by increasing the digital gain just prior to this re-quantization step, resulting in Figure 5.14, which had the best reported \( T_{sys} \).

5.7.2 On-Telescope

A total of five observation sessions were conducted. Each session began with a series of elevation/azimuth cuts (known as “Peak” scans) through a point source to ascertain correct pointing direction, a “Focus” scan, which varies the location of the feed along the axis of symmetry to verify proper placement of the feed. A calibration scan, either a series of elevation cuts or a “daisy”
Figure 5.13: Measured $T_{\text{sys}}$ with bit underflow.

Figure 5.14: Measured $T_{\text{sys}}$ with improved re-quantization.
trajectory around a point source, was then performed. Lastly, a survey of known neutral hydrogen and pulsars was then performed to ascertain the scientific capability of the instrument.

An acquired sensitivity map is shown in Figures 5.15 and 5.16. Like those in the July 2016 results, these maps were computed using (5.2). There are a few features in these maps worth noting. The first is that the maps are cut off at the higher elevation offset angles and that the highest sensitivity is not at the origin. This was the consequence of an erroneous pointing offset that was introduced by an incorrect interpretation of the recorded antenna position prior to the acquisition of the grid, which has since been corrected.

The second feature is the notch at the lowest elevation offsets in the Y-polarization map. This drop in sensitivity was caused by the exclusion of a problematic Y-polarized dipole from the beamformer. The exclusion of an element in the beamformer results in both a large loss of sensitivity in directions where the dish focal pattern (Airy pattern) is near that element and a small loss throughout the FoV.

Lastly, the sensitivity within the FoV is fairly uniform and is much more so than that shown in Figure 5.8, indicating that beamformer will perform equally well in any direction contained within the FoV. The peak sensitivities are 267.5 (X polarization) and 257.17 (Y polarization) m$^2$/K with corresponding normalized system temperatures $T_{sys}/\eta$ of 29.36 and 30.54 K. These temperatures rival that of the Green Bank L-Band single-pixel feed, which is approximately 27 K.

The normalized system temperatures $T_{sys}/\eta$ for the highest-sensitivity beam as a function of frequency are shown in Figures 5.17 and 5.18. As with the system temperature measurements from the OTF (see Section 5.7.1), at the lowest frequencies that are in the roll-off of the analog filter, $T_{sys}/\eta_{ap}$ appears to decrease dramatically. Since this behavior is similar to that of the OTF results, these temperature measurements are spurious and likely caused by bit underflow. As such, we ignore these values.

Measured element patterns are depicted in Figures 5.19 and 5.20. Note that in all of the following pattern plots, we have masked out regions that exhibited low sensitivity (i.e., the reported $T_{sys}/\eta$ was greater than 400 K).

Unfortunately, since the calibration grid was inadvertently cut off at the higher elevation offsets, the patterns for elements 8-9, and 18-19 are largely obscured, with the main lobe of the eighth element being completely missing. The patterns of elements in the entire outer ring (i.e.,
Figure 5.15: The X-polarization sensitivity map for a calibration grid on 3C295.

Figure 5.16: The Y-polarization sensitivity map for a calibration grid on 3C295.
Figure 5.17: X-polarization $T_{sys}/\eta$ spectrum on 3C295.

Figure 5.18: Y-polarization $T_{sys}/\eta$ spectrum on 3C295.
Figure 5.19: X-polarization element pattern maps from a calibration grid on 3C295.
Figure 5.20: Y-polarization element pattern maps from a calibration grid on 3C295.
elements 8-19) are not fully sampled either due to missing or insufficiently sensitive calibration points.

Formed beam patterns are shown in Figures 5.21 and 5.22, with contours drawn at 3 dB and 10 dB below peak levels. The main lobe for each beam is roughly circular, with a slight horizontal skew in beams three and five. Due to insufficient calibration points outside the FoV, the main lobes for beams 1-2, 6-7 are partially cut off, making analysis a challenge. Lastly, beams 6-7 for the Y-polarization are slightly distorted such that the main lobe appears pinched near a cross-elevation offset of zero, which is the result of the failed 14Y element. This missing element appears to have also raised the level of the first sidelobe of the seventh beam slightly.

5.8 August 2017 Commissioning Results

The third and final commissioning took place in August 2017, where each standalone operational mode (i.e., coarse-channel correlator, fine-channel correlator, and real-time beamformer) was tested. In between the May and August experiments, additional Y-polarized elements failed,
rendering the beamformer largely ineffective for that polarization, but X-polarized elements performed well. We calibrated the array across a wider angular region to obtain better element and beam pattern maps and make more calibration points available to constrain the formed beams. Resulting sensitivity maps for these calibration scans are shown in Figures 5.23 and 5.24.

The sensitivity map for the X-polarized elements is a very clean result: there is no cutoff of the FoV as in Figure 5.15, and every element is functioning well. Unfortunately, the same cannot be said of the map for the Y-polarized elements, where several elements (9Y, 14Y, and 15Y) failed or had unresolvable bit-alignment errors. Additionally, element 19Y was measured to be less sensitive, which could be the result of a failing LNA or power supply.

The corresponding measured $T_{sys}/\eta$ spectra for the highest-sensitivity beams are shown in Figures 5.25 and 5.26, where frequencies containing strong narrowband RFI have been masked out. For unknown reasons, the minimum system temperature was higher than in the May commissioning run: 36.08 K (X-polarization) and 39.64 K (Y-polarization), compared to 29.36 and 30.54 K respectively in May.
Figure 5.23: X-polarization sensitivity map for a calibration grid on 3C295.

Figure 5.24: Y-polarization sensitivity map for a calibration grid on 3C295.
Figure 5.25: X-polarization $T_{\text{sys}}/\eta$ spectrum on 3C295.

Figure 5.26: Y-polarization $T_{\text{sys}}/\eta$ spectrum on 3C295.
Measured element pattern maps are shown in Figures 5.27 and 5.28. As before, low-
sensitivity areas outside the designed FoV were masked out to make the images cleaner. In the
X-polarization pattern maps, we see much more beam structure, including the first and second
sidelobes, and, in some cases, the third sidelobe.

The Y-polarization maps, in contrast, are only useful for high-level diagnostics such as
confirming the array layout. One interesting feature in these plots, however, is that some element
patterns (e.g., 2Y and 7Y) are distorted. This could be the result of including the visibly noisier
element 19Y in the covariance matrices during array calibration, thus biasing the estimated steering
vectors in the region around 19Y’s main lobe.

There is unexpected asymmetry between elements 8X and 14X, which reside on opposite
ends of the array. Approximately half of the main lobe of 8X was well measured, but 14X appears
to have been poorly sampled for most of its main lobe. The fact that any element pattern is cut off
due to insufficient sensitivity is puzzling, but this asymmetry could be indicative of a misalignment
of the array within the cryogenic dewar such that some elements are being obscured by the edges
of the dewar opening.

Figures 5.29 and 5.30 show measured beam patterns for the seven beam locations specified
in Figure 5.11. Since the calibration procedure was performed over a larger set of angles, we
can now resolve the entire 3 dB contour of each formed beam. We can also see parts of the
second sidelobe in the steered beams. Again, the Y-polarization beams are not well formed due to
many missing elements, but the center beam does appear to be approximately circular and may be
salvageable.

To illustrate the utility of a larger calibration set, we attempted to constrain the first null
of each beam to be placed 15 arcminutes (0.25 degrees) away from the beam main lobe. This
is similar to the work done by Elmer to place nulls in the spillover region of the dish [31], [69].
Our approach was to first collect estimated steering vectors that corresponded to angles that were
closest, in a Euclidean sense, to the desired constraint locations, as shown in Figure 5.31. Any
obtained steering vectors whose angle was more than one-tenth of a beamwidth away from the
specified constraint circle were not used. We also discarded every steering vector whose sensitivity
was low. With these two pruning criteria, every steered beam was left with an insufficient number
of steering vectors to fully constraint the null.
Figure 5.27: X-polarization element pattern maps from a calibration grid on 3C295.
Figure 5.28: Y-polarization element pattern maps from a calibration grid on 3C295.
Figure 5.29: X-polarization formed beam maps from a calibration grid on 3C295.

Figure 5.30: Y-polarization formed beam maps from a calibration grid on 3C295.
The number of constraint points far exceeds the available number of degrees of freedom for the beamformer. We therefore use a two-step approach to forming the constraint matrix by finding a reduced-rank representation of the null-constraint matrix \([111]\) and concatenating the main lobe constraint vector, or

\[
\mathbf{C}^* = [\mathbf{a}(\theta_{\text{center}})]\mathbf{\tilde{C}},
\]  

where \(\mathbf{a}(\theta_{\text{center}})\) is the steering vector corresponding to the desired beam steering angle, and

\[
\mathbf{C} = \mathbf{USV}^H,
\]  

\[
= [\mathbf{U}_1|\mathbf{U}_2] \begin{bmatrix} \mathbf{S}_1 & 0 \\ 0 & \mathbf{S}_2 \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^H \\ \mathbf{V}_2^H \end{bmatrix}
\]  

\[
\mathbf{C}_1 = \mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^H.
\]  

The columns of \(\mathbf{C}\) are the null constraint steering vectors, and \(\mathbf{USV}^H\) is the singular value decomposition of \(\mathbf{C}\). The matrix \(\mathbf{C}_1\) is a reduced-rank approximation of \(\mathbf{C}\), where the columns of \(\mathbf{U}_1\) are the \(Q\) left singular vectors corresponding to the \(Q\) largest singular values, \(\mathbf{S}_1\) is a diagonal matrix whose entries are the \(Q\) largest singular values, and the columns of \(\mathbf{V}_1\) are the \(Q\) right singular
vectors. Thus the constraint conditions for the LCMV formulation from (2.21) becomes

\[
\begin{align*}
C^H w &= f \\
V^H S^H U^H w &= f \\
S^H U^H w &= V^H f \\
\tilde{C}^H w &= \tilde{f}.
\end{align*}
\] (5.11)

Only the steering vectors corresponding to the null constraint are used to form this reduced-rank representation so as to guarantee that the main lobe constraint is met.

Figure 5.32 shows the resulting beam patterns when \( Q = 6 \), where we have superimposed a white dashed circle to represent the desired null locations. Clearly, the central beam pattern exhibits the desired null, but the steered beams appear largely unaffected.

We can improve the patterns of the steered beams by increasing \( Q \) to be 10, and the resulting beam patterns are shown in Figure 5.33. Again, the central beam exhibits the null very well, and
now most of the steered beams now have a clear null in the desired locations. Beams 6-7, however, only partially form a null in the specified directions, indicating that the array steering vectors are not well represented by a low-rank approximation.

Lastly, we set $Q$ to 15 and form beamformer weights again, resulting in Figure 5.34. Here the null is much more apparent in all of the beams, but this has also resulted in a much higher first sidelobe. This illustrates the natural trade-off that LCMV suffers from: fewer degrees of freedom results in a less controlled beam shape that minimizes noise variance overall, and more degrees of freedom results in a more controlled pattern at the expense of increasing overall noise variance. This method would be more effective on larger PAFs like the 188-element ASKAP array and the proposed 160-element Advanced L-Band Phased Array Camera for Arecibo (ALPACA) instrument.

We also acquired a series of calibration points for the center of the array while varying the RF center frequency. This enabled us to survey the overall sensitivity of the array for any possible operating frequency, and the resulting $T_{sys}/\eta$ spectra are shown in Figures 5.35 and 5.36, where spurious narrowband spikes were masked out. We have also superimposed a 10th order polynomial
fitted curve. According to the fitted model, the minimum achieved $T_{\text{sys}}/\eta$ was 35.93 and 35.09 K for X and Y polarizations respectively. The 1 dB bandwidth (i.e., the bandwidth in which the noise temperature is less than a 1 dB increase from the minimum) was 267 MHz and 238 MHz.

5.9 Conclusions and Future Work

There are still some results that are pending processing from our astronomer colleagues, including a beamformed map of a faint HI source and a detection of a pulsar using every beam of the real-time beamformer. Furthermore, there were some lingering issues with the system that need to be resolved. These include (1) an undesirable “scalloping” spectral structure that appears in the fine-channel data that is the result of multi-stage channelization (a solution has been identified and is currently being implemented), (2) an as-of-yet unexplained bug or hardware limitation that at-times increases latency in the pipeline and dropping packets, and (3) the missing implementation of the correlator and beamformer for concurrent operation.
Figure 5.35: X-polarization $T_{sys}/\eta$ broad spectrum on 3C123.

Figure 5.36: Y-polarization $T_{sys}/\eta$ broad spectrum on 3C123.
Overall, the FLAG instrument will be a reliable beamforming array system for the Green Bank Telescope moving forward. It will require some non-trivial work to integrate the remaining functionality and fully integrate it into the Green Bank control system for common-user use, and staff at the telescope site are showing interest in doing so.
CHAPTER 6. DEEP BROAD NULL FORMATION VIA ORTHOGONAL PROJECTION AND SUBSPACE TRACKING

6.1 Introduction

Radio frequency interference (RFI) is an ever-growing problem in many applications including radio astronomy [112], [113]. Deep-space sources of interest (SOIs) exhibit signal-to-noise ratios (SNRs) that can be incredibly low (e.g., -30 dB or lower), which makes these signals extremely sensitive to RFI. In such situations, weak interference that is below the noise floor can still corrupt the SOI and require a zero-forcing mitigation approach.

Phased array feed (PAF) antenna systems, in addition to their ability to widen the field of view of a large reflector dish and to synthesize fine-resolution interferometric images, can also combat the growing RFI problem. Projection operators can place a pattern null in the direction of the RFI to spatially filter the interference. However, existing techniques are insufficient in low-SNR scenarios. They are either unable to sufficiently attenuate the RFI to levels far below the noise floor, are highly susceptible to the effects of RFI motion, or lack necessary knowledge of the array manifold (i.e., a steering vector or spatial signature vector) to constrain the beam or project out the RFI subspace. In this work, we aim to solve these problems with a new zero-forcing algorithm that can mitigate in-motion RFI using orthogonal projections, thus forming a spatially broad null using \textit{in-situ} short-time estimates of the array steering vector. The short-term nature of the estimates minimizes the usual manifold estimation errors caused by motion.

A traditional approach to RFI mitigation in beamforming systems is to use linear-constraint minimum variance (LCMV) beamforming to minimize overall variance subject to carefully placed null and beam pattern constraints [68]. These constraints, however, require accurate \textit{a priori} knowledge of the array manifold (at least in the constraint directions), which may not be knowable beforehand since RFI often arrives from directions which are in the deep sidelobes of a high gain dish pattern. Furthermore, variance-minimizing beamformers without null constraints are less ef-
A time-averaged two-dimensional vector representation of stationary RFI and SOI. Here, we project the signal vector $\mathbf{x}$ onto the orthogonal complement of the RFI steering vector $\sigma x a(\theta_i)$. (b) A time-averaged two-dimensional vector representation of moving RFI and stationary SOI. Here, the RFI subspace is not rank-one due to motion over the observation window, making a unit-rank subspace orthogonal complement of the RFI steering estimate ineffective.

Figure 6.1: An example of subspace smearing.

effective in radio astronomy canceling scenarios in low-SNR environments due to their inability to drive weak interference far below the noise floor. Therefore a zero-forcing orthogonal projection operator, which projects the sensor data onto a subspace that is orthogonal to the RFI, is preferred.

Without *a priori* knowledge of the array manifold for the angular region of the RFI, one must estimate the RFI spatial signature *in situ*, which involves the extraction of dominant eigenvectors from a spatial sample correlation matrix [114] to form a matrix operator that projects the RFI onto its orthogonal complement [18], [56], [60], as shown in Figure 6.1a. This simple procedure is greatly complicated when the RFI is in motion, which causes the interferer’s spatial signature to evolve over time yielding a higher dimensional RFI subspace, as seen in Figure 6.1b. This phenomenon is known as subspace smearing [61], [62].

One way to compensate for subspace smearing is to compute spatial sample correlation matrices and corresponding projection operators over shorter time intervals in which the RFI is approximately stationary [61], [62], but this results in higher estimation error in the sample correlation matrix and, consequently, in the RFI subspace estimate. As a result, one must balance two conflicting requirements: (1) long integration lengths for low sample estimation error, and (2) short integration lengths to prevent subspace smearing effects. Additionally, shorter-time intervals result in higher data rates from a correlator, thus increasing the computational requirements for real-time processors. Another approach to mitigating RFI motion is to estimate the subspace using more de-
degrees of freedom from a long-term estimate of the spatial correlation matrix to form a broad null, but this has variable effectiveness depending on interference power and can result in unintended signal cancelation.

In this chapter, we propose a solution to mitigating in-motion RFI that both minimizes sample estimation error and projects out the multi-dimensional structure of the moving RFI for varying power levels by forming a spatially broad null. This is done by extracting dominant components out of a series of short-time, approximately stationary, rank-one RFI subspace estimates and using these extracted components to form the projection operator. The short-term nature of the individual estimates ensures RFI stationarity in each estimate, and the aggregated estimates preserve the multi-dimensional subspace structure of the RFI induced by motion over the long-term. Furthermore, this long-term aggregated subspace estimator performs additional integration thus driving down sample estimation error.

The concept of a broad null was first introduced by Er who solved a quadratic constraint optimization problem to force the beam response of a uniform line array (ULA) to be below a given threshold in the neighborhood of the RFI [115]. Baranoski generalized the problem for non-ULAs and proposed an orthogonal projection method using a reduced-rank representation of the array manifold across a continuous trajectory in which the RFI could arrive [116]. Li et al. demonstrated that only discrete samples of the array manifold across said trajectory are necessary to create a reduced-rank representation for the RFI region [117]. Hossain et al. developed techniques for robust constrained broadband pattern synthesis that can place broad nulls at prescribed locations without a projection operator [118]. These works indicate that one can form broad nulls easily if one has a priori knowledge of the array manifold. The proposed algorithm relaxes this constraint and permits broad null formation in completely uncalibrated regions of the array, as seen, for example, with directional elements in the array where RFI is observed through scattering, diffraction, or other deep sidelobe regions of the antennas.

The performance of orthogonal projections can also be improved for low-power interference by incorporating an RFI-tracking auxiliary antenna as proposed by Jeffs et al. [57], [58]. Sardarabadi et al. demonstrated that if the auxiliary is equipped with a PAF and a real-time adaptive beamformer, one can achieve even greater interference cancelation [59]. Both of these approaches
enhance the performance of the projection operator but increase the computational resources and costs required to adopt them.

We begin by introducing a general discrete-time signal model for antenna array and single-antenna complex voltages and then introduce projection-based RFI mitigation theory. Next, we will examine two existing approaches to estimating the RFI subspace. The first method neglects motion and provides maximum integration times, while the second, more traditional approach sacrifices lower sample estimation error in order to compensate for motion. We then introduce our broad null algorithm and demonstrate how it is able to mitigate motion effects without suffering from higher sample estimation error. Lastly, we will present simulation results for a PAF on the Green Bank Telescope (GBT).

6.2 Signal Model and Algorithm Development

Consider an array of $M$ antenna array elements that observes an SOI, a single interferer, and noise. The complex basebanded voltages in a narrowband frequency channel are given by

$$x[n] = s[n]a(\theta_s) + i[n]a(\theta_i[n]) + z[n],$$

where $s[n]$ and $i[n]$ represent the voltage samples of the desired signal and interference sources respectively. For radio astronomy, these are modeled as zero-mean, temporally white circular complex Gaussian random processes with respective variances $\sigma_s^2$ and $\sigma_i^2$. The vector $a(\theta)$ represents the deterministic array response to a plane wave from the two-dimensional arrival angle $\theta$ and is also referred to as a steering vector or spatial signature vector. Without loss of generality, we assume that the first element of $a(\theta)$ is real. The vector process $z[n]$ represents the array noise, which is a zero-mean, temporally white, circular complex Gaussian random vector process with spatial correlation $R_z$. The SOI and interfering source’s angles of arrival are denoted as $\theta_s$ and $\theta_i[n]$ respectively, where the RFI angle is time-varying due to motion. While (6.1) is for a single point-source interferer scenario, the following development is easily generalized to multiple RFI sources through superposition [58]. Also note that this model only differs from (2.1) in that the RFI angle is now time-varying.
The statistics of \( x[n] \) are non-stationary, necessitating the need for a time-varying correlation matrix model. The sample correlation matrix of \( x[n] \) in an \( N \)-sample window starting from time sample \( n \) is

\[
\hat{R}_{x,N}[n] = \frac{1}{N} \sum_{k=0}^{N-1} x[n+k]x^H[n+k],
\]

(6.2)

and the expectation of which is

\[
R_{x,N}[n] = E\{\hat{R}_{x,N}[n]\},
\]

(6.3)

\[
= \frac{1}{N} \sum_{k=0}^{N-1} E\{x[n+k]x^H[n+k]\},
\]

(6.4)

\[
= \frac{1}{N} \sum_{k=0}^{N-1} R_x[n+k],
\]

(6.5)

where \( E\{\cdot\} \) is the expectation operator, and, assuming statistical independence between the SOI, RFI, and noise,

\[
R_x[n] = E\{x[n]x^H[n]\},
\]

(6.6)

\[
= R_s + R_{i,N}[n] + R_z.
\]

(6.7)

We therefore proceed with an \( N \)-sample-long window total array correlation model given by

\[
R_{x,N}[n] = \frac{1}{N} \sum_{k=0}^{N-1} E\{x[n+k]x^H[n+k]\},
\]

(6.8)

\[
= R_s + R_{i,N}[n] + R_z,
\]

(6.9)

where \( R_{i,N}[n] \) depends on \( n \) due to RFI motion, and

\[
R_s = \sigma_s^2 a(\theta_s)a^H(\theta_s),
\]

\[
R_{i,N}[n] = \frac{\sigma_i^2}{N} \sum_{k=0}^{N-1} a(\theta_i[n+k])a^H(\theta_i[n+k]).
\]

(6.9)
Now consider a single RFI-tracking auxiliary antenna that is synchronously sampled with the array voltages $x[n]$ to obtain a higher interference-to-noise ratio (INR) copy of the RFI for lower estimation error of RFI parameters and deeper nulls in the case of low INR RFI [57], [58]. The auxiliary complex basebanded voltage in the same narrowband channel is given by

$$x_a[n] = s[n]a(\theta'_s[n]) + i[n]a(\theta'_i) + z_a[n],$$  \hspace{1cm} (6.10)

where $a(\theta)$ is the auxiliary antenna gain and phase response (relative to the first element of $a(\theta)$) to a plane wave arriving from the two-dimensional angle $\theta$, $\theta'$ represents the angle of arrival in the coordinate system of the auxiliary sensor, and $z_a[n]$ is the auxiliary antenna noise, which is a white, zero-mean circular Gaussian random process with variance $\sigma^2_z$. Note that the SOI arrival angle at the auxiliary antenna is time-varying since it tracks the RFI, inducing relative SOI motion.

The array and auxiliary voltages can be stacked column-wise as follows:

$$y[n] = \begin{bmatrix} x[n] \\ x_a[n] \end{bmatrix}.$$  \hspace{1cm} (6.11)

The sample correlation matrix for this new vector process is given by

$$\hat{R}_{y,N}[n] = \frac{1}{N} \sum_{k=0}^{N-1} y[n+k]y^H[n+k],$$  \hspace{1cm} (6.12)

and the time-average correlation matrix model for this new vector process is given by

$$R_{y,N}[n] = E\{\hat{R}_{y,N}[n]\}$$
$$= \frac{1}{N} \sum_{k=0}^{N-1} E\{y[n+k]y^H[n+k]\}$$
$$= \begin{bmatrix} R_{x,N}[n] & r_{xa,N}[n] \\ r_{xa,N}^H[n] & \sigma^2_{a,N}[n] \end{bmatrix},$$  \hspace{1cm} (6.13)
where
\[
\mathbf{r}_{xa, N}[n] = \frac{1}{N} \sum_{k=0}^{N-1} E \{ \mathbf{x}[n+k] \mathbf{x}^*_{a}[n+k] \}, \quad (6.14)
\]
\[
\sigma_{a, N}^2[n] = \frac{1}{N} \sum_{k=0}^{N-1} E \{ |\mathbf{x}_{a}[n+k]|^2 \}. \quad (6.15)
\]

### 6.3 Projection-Based RFI Mitigation

#### 6.3.1 Background

In projection-based RFI mitigation, the goal is to estimate the RFI subspace \( \mathbf{U} \) for \( \mathbf{a}(\theta_i[n]) \) in an \( N \)-sample-long window such that
\[
\mathbf{U} = \text{span}\{ \mathbf{a}(\theta_i[n]), 0 \leq n \leq N-1 \}. \quad (6.16)
\]

We can then form the projection \( \mathbf{P} = \mathbf{I} - \mathbf{U}(\mathbf{U}^H \mathbf{U})^{-1} \mathbf{U}^H \), which projects \( \mathbf{x}[n] \) onto the orthogonal complement of \( \mathbf{U} \). \( \mathbf{P} \) can then be used in a variety of array processing algorithms to mitigate the RFI (see Section 2.8.1 for more details). Recognizing that \( \mathbf{P} \) may be applied somewhat differently in various applications and algorithms, we focus on calculating \( \tilde{\mathbf{R}}_x \), the RFI-cleaned spatial covariance matrix, as one would for interferometric synthesis imaging arrays [119] for consistency in the following discussion.

The estimate of the RFI subspace \( \mathbf{U} \) is typically given by the dominant eigenvectors of sample spatial correlation matrices [114]. This approach works when \( \text{trace}\{\mathbf{R}_i\} \gg E \{ \mathbf{R}_s \} \) and either \( \text{trace}\{\mathbf{R}_i\} \gg E \{ \mathbf{R}_z \} \) or \( \mathbf{z}[n] \) is spatially white so \( \mathbf{R}_z = \sigma_z^2 \mathbf{I} \). In radio astronomical applications, often neither of these last two conditions are true. Use of an RFI-tracking auxiliary antenna and noise-whitening procedures relax these requirements and thus improve performance.

The number of samples used in computing a sample correlation matrix is a critical parameter in the performance of the projection. For example, if one does nothing to compensate for RFI motion, the integration length can be large in order to minimize sample estimation error (denoted hereafter as the “motion-agnostic” approach). If one attempts to compensate for RFI motion, the
Figure 6.2: A motion-agnostic approach to RFI mitigation that is ineffective when the RFI is moving quickly. Depending on the application, the desired data product is either the RFI-cleaned correlation $\hat{R}_x$ (e.g., for synthesis imaging) or projection operator (e.g., for beamformer weight calculation and updates).

Integration length should be relatively small in order to guarantee that the underlying RFI subspace is stationary to avoid subspace smearing (denoted hereafter as the “motion-cognizant” approach). This trade-off between sample estimation error and motion smearing is not present when utilizing our proposed broad null method. We will illustrate how the motion-agnostic and motion-cognizant approaches are implemented and then introduce our alternative approach.

6.3.2 A Framework for Existing RFI Projection Algorithms

Motion-Agnostic Approach

Figure 6.2 depicts an approach that does not compensate for the motion of the RFI during an $N$-sample long-term integration (LTI) window. Interpreting over an LTI that is long relative to the environment stationarity time (due to RFI motion) does reduce noise-induced sample estimation error but poorly tracks the evolution of the RFI signal component. In this approach, we compute the sample correlation matrix $\hat{R}_x$ across $N$ time samples as defined in (6.2). Since $\hat{R}_x$ is Hermitian, it can be decomposed as

$$\hat{R}_x = U\Lambda U^H,$$  \hspace{1cm} (6.17)

$$U = [U_1 | U_2],$$  \hspace{1cm} (6.18)

where $U$ is a unitary matrix whose columns are the eigenvectors of $\hat{R}_x$, $\Lambda$ is a diagonal matrix with entries being the sorted eigenvalues in descending order, the columns of $U_1$ are the $Q$ dominant
A motion-agnostic approach when augmented with an auxiliary antenna that tracks the RFI. (1) indicates that only the matrix entries corresponding to the array correlations (i.e., \( \hat{R}_x \) from (6.13)) are propagated on this line, and (2) indicates that only the correlations between the array and the auxiliary antenna (i.e., \( \hat{r}_{xa} \) from (6.13)) are propagated on this line.

eigenvectors and are our estimate of the RFI subspace when the RFI is strong relative to the SOI, and the columns of \( U_2 \) are the remaining eigenvectors. The value \( Q \) is a design parameter that indicates the hypothesized RFI subspace dimension size, which is increased by RFI motion and the number of interfering sources. Thus, the projection operator is

\[
P = I - U_1 U_1^H. \tag{6.19}
\]

When the array is augmented with an RFI-tracking auxiliary antenna, the approach is modified slightly, as shown in Figure 6.3. We extract the entries of \( \hat{R}_y \) corresponding to \( \hat{r}_{xa} \) (see (6.13)), which is our rank-one estimate for the RFI subspace to form the projection matrix

\[
P = I - \hat{r}_{xa} (\hat{r}_{xa}^H \hat{r}_{xa})^{-1} \hat{r}_{xa}^H. \tag{6.20}
\]

With or without an auxiliary antenna, this algorithm places a null somewhere in the angular neighborhood of the RFI resulting in some interference suppression. Furthermore, given the long integration length, the estimation error variance due to noise alone is low. However, the true underlying RFI subspace is not rank-one due to subspace smearing, resulting in residual interference power bypassing the projection operation. Including additional eigenvectors (i.e., \( Q > 1 \)) does reduce this residual interference power but does not do so reliably across varying interference power levels, as will be demonstrated in Section 6.4.

The long integration length is also appealing since the data rate coming out of the augmented correlator is small. Assuming one saves only the non-redundant lower-triangular entries of
Figure 6.4: Traditional solution to moving RFI mitigation that trades off the integration length of the sample correlation matrix to mitigate motion at the expense of increased sample estimation error. (1) indicates that only the matrix entries corresponding to the array correlations (i.e., $\hat{R}_x$ from (6.13)) are propagated on this line, and (2) indicates that only the correlations between the array and the auxiliary antenna (i.e., $\hat{r}_{xa}$ from (6.13)) are propagated on this line.

The Hermitian correlation matrix $R_y$, the resulting data rate, in bits-per-second, is given by

$$d = K_c \frac{M(M + 1)}{2T_{LTI}} b,$$

where $K_c$ is the number of processed frequency channels, $T_{LTI}$ is the length of the integration window in seconds, and $b$ is the bit-width of each correlation entry. Thus, for a system like the Focal L-Band Array for the GBT (FLAG) [120]–[122], which processes 38 elements, 500 coarse frequency channels, and 64 bits per matrix entry, the correlator output data rate is approximately 23.7 Mega-bits-per-second (Mbps) for $T_{LTI} = 1$s. For a single PAF-equipped dish on the Australian Square Kilometre Array Pathfinder (ASKAP) instrument [43], [44], which has 188 elements, 64 coarse frequency channels, and 64 bits per matrix entry, the data rate would be approximately 72.8 Mbps.

**Motion-Cognizant Approach**

The conventional approach to estimating the RFI subspace in the presence of motion with an auxiliary antenna is shown in Figure 6.4. An “unassisted” version (not shown) is similar to that in Figure 6.2, but it does not utilize an auxiliary antenna and uses dominant eigenvectors as the RFI subspace estimate. In both cases, a series of $L$ correlation matrices $\hat{R}_{y,l}$ ($\hat{R}_{x,l}$ for unassisted case), $0 \leq l \leq L - 1$ are estimated as in (6.2) and (6.12) over a short-time integration (STI) window using $K$ time samples ($N = KL$) each. We then obtain a rank-one estimate of the RFI subspace from the
th STI by either extracting the dominant eigenvectors $U_{1,l}$ (no auxiliary) or the cross-correlation vector $r_{xa,l}$ (with auxiliary), where we have introduced the subscript $l$ to indicate that the estimates are from $\hat{R}_{y,l}$ (or $\hat{R}_{x,l}$ for the unassisted case). A projection matrix $P_l$ is then formed using $r_{xa,l}$ (or $U_{1,l}$ without an auxiliary) for each STI window as in (6.19) and (6.20). In a real-time beamforming application, the weights must be updated every STI. In a post-correlation beamforming or interferometric synthesis imaging application, an RFI-reduced LTI sample correlation matrix is then formed from many “cleaned” STI windows yielding

$$\tilde{R}_x = \frac{1}{L} \sum_{l=0}^{L-1} P_l \hat{R}_{x,l} P_l^H.$$  \hspace{1cm} (6.22)

This approach assumes that the RFI subspace is rank-one over a short window of length $K$, which allows for better motion cancelation. The short time span of the window, however, results in higher sample estimation error in the correlation matrix and, thus, poorer cancelation. Furthermore, this approach requires significantly more computational resources than the motion-agnostic approach since it requires the computation of $L$ sample correlation matrices and corresponding projection matrices $P_l$ must be computed instead of one each.

The exact correlator output data rate for this approach depends on the STI length. The appropriate length depends largely on the speed of the moving RFI, array geometry, baseline length, and processing bandwidth, but Leshem et al. recommends a very small window length on the order of 10 ms for long baseline interferometers [55], [56], [59], [123]. For $\hat{R}_y$, this yields an effective output data rate of

$$d = K_e \frac{M(M + 1)}{2T_{STI}} b,$$  \hspace{1cm} (6.23)

where $T_{STI}$ is the STI length in seconds. For FLAG the data rate is 2.624 gigabits-per-second, and for a single ASKAP dish it is 7.277 gigabits-per-second.
6.3.3 Proposed Algorithm for Broad Null Formation

Algorithm Description

We propose the algorithm shown in Figures 6.5 (with no auxiliary antenna) and 6.6 (with auxiliary antenna) as alternative solutions, which offer both better motion mitigation and lower sample estimation error, thus producing deeper, spatially broader nulls.

Again, the goal of projection-based RFI mitigation is to estimate the RFI subspace $U$ across the entire integration window $0 \leq n \leq N - 1$ such that

$$U = \text{span}\{a(\theta[n]), 0 \leq n \leq N - 1\}.$$
If the array steering vector is known for all $\theta[n], 0 \leq n \leq N - 1$, the RFI subspace $\mathbf{U}$ can be found through an eigenvalue decomposition of [117]

$$
\mathbf{R}_A \triangleq \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{a}(\theta[n])\mathbf{a}^H(\theta[n]).
$$

(6.24)

When not assisted by an auxiliary antenna, we can express the evolving steering vector of a point-source RFI which has high power relative to that of the SOI via the Karhunen-Loève theorem, which states

$$
\mathbf{a}(\theta[n])\mathbf{a}^H(\theta[n]) \approx \lambda_1[n] \mathbf{u}_1[n] \mathbf{u}_1^H[n],
$$

(6.25)

where $\lambda_1[n]$ is the largest eigenvalue of $\mathbf{R}_x[n]$ (see eq. (6.6)) and $\mathbf{u}_1[n]$ is the corresponding eigenvector. Thus, the time-averaged matrix corresponding to the RFI over an $N$-sample window is approximately

$$
\mathbf{R}_A \approx \frac{1}{N} \sum_{n=0}^{N-1} \lambda_1[n] \mathbf{u}_1[n] \mathbf{u}_1^H[n],
$$

(6.26)

$$
\triangleq \mathbf{R}_U.
$$

(6.27)

An estimate for this matrix is computed on a per-STI basis such that

$$
\hat{\mathbf{R}}_U = \frac{1}{L} \sum_{l=0}^{L-1} \lambda_{1,l} \mathbf{u}_{1,l} \mathbf{u}_{1,l}^H,
$$

(6.28)

where $\lambda_{1,l}$ is the largest eigenvalue of $\hat{\mathbf{R}}_x,l$, and $\mathbf{u}_{1,l}$ is the corresponding eigenvector. We can then perform an eigenvalue decomposition of this new matrix to extract the RFI subspace $\mathbf{U}$ such that

$$
\hat{\mathbf{R}}_U = [\mathbf{U}_1 | \mathbf{U}_2] \Lambda \left[ \begin{array}{c} \mathbf{U}_1^H \\ \mathbf{U}_2^H \end{array} \right],
$$

(6.29)

where the columns of $\mathbf{U}_1$ are the $Q$ dominant eigenvectors of $\hat{\mathbf{R}}_U$, and the columns of $\mathbf{U}_2$ are the remaining eigenvectors. Recall that $Q$ is a design parameter to select the dimensionality of the subspace to be removed by the projection operator $\mathbf{P}$. For a single moving interferer and
appropriate LTI lengths, we have found $Q = 2$ or $3$ performs well. Using $U_1$ as an estimate for the RFI subspace across the entire LTI, we obtain the projection

$$P = I - U_1 U_1^H.$$  \hfill (6.30)

This approach can be expanded to $J$ RFI sources by including more scaled eigenvectors from $\hat{R}_{x,l}$, so (6.28) becomes

$$\hat{R}_U = \frac{1}{L} \sum_{l=0}^{L-1} \sum_{j=1}^{J} \lambda_{j,l} u_{j,l} u_{j,l}^H.$$  \hfill (6.31)

We then decompose $\hat{R}_U$ as in (6.29), but extract $JQ$ eigenvectors instead of only $Q$.

When utilizing an RFI-tracking auxiliary antenna, we use the cross-correlations between the primary array and the auxiliary antenna or

$$r_{xa}[n] = E\{x[n]x_a^*[n]\}$$
$$= \sigma_s^2 a(\theta_s) a^*(\theta_s'[n]) + \sigma_i^2 a(\theta_i[n]) a^*(\theta_i'[n]) + r_{za}, \quad (6.32)$$

where $r_{za} = E\{z[n]z_a^*[n]\}$. The gain of the auxiliary (embedded in $a(\theta)$), dramatically attenuates the SOI and amplifies the RFI, allowing for the approximation

$$r_{xa}[n] \approx a(\theta_i[n]) a^*(\theta_i'). \quad (6.33)$$

In this treatment, we assume that $\theta_i'$ is constant over time. We can then write

$$R_A = \frac{1}{N} \sum_{n=0}^{N-1} a(\theta[n]) a^H(\theta[n]),$$
$$\approx \frac{1}{N} \sum_{n=0}^{N-1} r_{xa}[n] r_{xa}^H[n], \quad (6.34)$$
$$= R_{xa}. \quad (6.35)$$
As with $\mathbf{R}_U$, an estimate for this matrix is computed on a per-STI basis, or

\[
\hat{\mathbf{r}}_{xa,l} = \frac{1}{K} \sum_{n=0}^{K-1} x[n + Kl] x^*_q[n + Kl]
\]  
(6.36)

\[
\hat{\mathbf{R}}_{xa} = \frac{1}{L} \sum_{l=0}^{L-1} \hat{\mathbf{r}}_{xa,l} \hat{\mathbf{r}}^H_{xa,l},
\]  
(6.37)

and this matrix can be also decomposed, as in (6.29), and the projection matrix is then formed using $\mathbf{U}_1$ as in (6.30).

The auxiliary-assisted approach is trivially expanded to cancel multiple RFI sources by including an additional RFI-tracking antenna per each additional RFI source as shown in Figure 6.6. Eq. (6.37) then becomes

\[
\hat{\mathbf{R}}_{xa} = \frac{1}{L} \sum_{l=0}^{L-1} \sum_{j=1}^{J} \hat{\mathbf{r}}_{xa,j,l} \hat{\mathbf{r}}^H_{xa,j,l},
\]  
(6.38)

where $\hat{\mathbf{r}}_{xa,j,l}$ are the cross correlations between the array and the $j$th auxiliary antenna for the $l$th STI window.

**Discussion**

The broad null approach retains the advantages of both the motion-agnostic (i.e., low sample estimation error) and motion-cognizant (i.e., effective moving RFI cancelation) approaches at the cost of using up a few additional degrees of freedom (i.e., $\mathbf{P}$ is lower rank). The higher sample estimation error in the motion-cognizant approach is caused by insufficient integration, and the poor motion cancelation in the motion-agnostic approach is caused by averaging the motion of the RFI over an LTI. The broad null’s ability to reduce sample estimation error and improve moving RFI cancelation is rooted in the unique matrices $\hat{\mathbf{R}}_U$ and $\hat{\mathbf{R}}_{xa}$ that are created by averaging the outer product of successive short-time estimates of the RFI steering vector. For example,

\[
\hat{\mathbf{R}}_{xa} = \frac{1}{L} \sum_{l=0}^{L-1} \hat{\mathbf{r}}_{xa} \hat{\mathbf{r}}^H_{xa}.
\]
Here we observe that this step performs additional integration in outer-product form, which drives down the RFI subspace sample estimation error. Furthermore, this approach preserves the multi-rank nature of the moving RFI subspace over the full LTI and, through an eigenvalue decomposition, yields an orthonormal basis for the subspace, thus forming a broad null.

Lastly, compared to the motion-cognizant approach, the broad-null auxiliary-assisted architecture provides a lower correlator output data rate of

\[ d = K_c \left( \frac{M}{T_{STI}} + \frac{M(M-1)}{2T_{LTI}} \right) b, \]  

(6.39)

where the \( \frac{M}{T_{STI}} \) term is the data rate contribution from an auxiliary cross-correlator as defined in (6.36), and the \( \frac{M(M-1)}{2T_{LTI}} \) term is the contribution from an array correlator as defined in (6.2). Assuming a one-second LTI window length, the resulting correlator output data rate for FLAG is 1.441 gigabits-per-second, and the rate for ASKAP is 0.149 gigabits-per-second, a reduction of 45% and 98% respectively compared to the conventional, motion-cognizant approach in (6.23).

In summary, the broad null approach reduces sample estimation error, effectively tracks RFI motion to produce deep nulls, and reduces overall computational burden making it more cost-effective than existing methods.

### 6.4 Simulation

#### 6.4.1 Simulation Details

Our simulation code is designed to represent the processing chain used for a typical array-enabled radio astronomy observation using a large parabolic reflector, where the SOI is modeled as an extremely weak, temporally white, Gaussian random process, which can only be reliably detected by subtracting “off” power estimates from “on” power estimates [18]. As such, the data products used are time-averaged post-beamformed powers in the case of PAF astronomy and integrated sample cross-correlation matrices for synthesis imaging arrays [119].

We generated spatial steering vectors for the GBT instrument using a sophisticated reflector model and derived fields and impedances using HFSS, a commercial antenna modeling and electromagnetic wave simulation software suite. This code factored in several array-sensitive pa-
rameters such as inter-element mutual coupling, active impedance matching, LNA noise and its recoupling, antenna loss, and spillover noise [66]. The modeled SOI was Messier 5 (M5), which is a radiating globular cluster of stars, and the RFI source was the orbiting GPS BIIR-19 satellite downlink at 1550 MHz. Both were simulated for 21 seconds starting at Julian date 2457618.39406. The orbital tracks for actual satellite transits on this date were modeled using the XEphem software package from the Clear Sky Institute with updated positions every second. These position samples were then interpolated to millisecond resolution. The SNR was fixed at -29.5 dB, which is realistic for M5 and a large cryogenically cooled antenna, and the interference power was varied.

Power levels were defined post-beamforming such that

\[
P_s = \mathbf{w}^H \mathbf{R}_s \mathbf{w} = \sigma_s^2 |\mathbf{w}^H \mathbf{a}(\theta_s)|^2 \tag{6.40}
\]

\[
P_i = \mathbf{w}^H \mathbf{R}_{i,N}[n] \mathbf{w} = \frac{\sigma_i^2}{N} \sum_{k=0}^{N-1} |\mathbf{w}^H \mathbf{a}(\theta_{i[n+k]})|^2 \tag{6.41}
\]

\[
P_n = \mathbf{w}^H \mathbf{R}_n \mathbf{w}, \tag{6.42}
\]

where \(\mathbf{w}\) is the beamformer weight vector per STI or LTI as appropriate, \(P_s, P_i, \) and \(P_n\) correspond to the signal, RFI, and noise beamformed power respectively.

The three orthogonal projection architectures described above were simulated. We performed a total of 50 Monte Carlo trials for each algorithm, where the results from the entire 21-second scan were averaged together. We used an LTI length of one second and an STI length of 10 milliseconds.

A 19-element dual-polarization thickened flared dipole hexagonal array feed was simulated with 0.7\(\lambda\) spacing (shown in Figure 6.7). This array was designed to provide peak sensitivity for the optics of the GBT [124]. These parameters are consistent with the FLAG PAF system we recently commissioned on the GBT. A bandwidth of 303 kHz was used, matching the coarse channel bandwidth for FLAG.

Lastly, these results have also undergone what is called subspace bias correction, where the effects of successive projection operators are corrected on-average in order to restore the quiescent,
or unmodified beam pattern [56] (see Section 2.8.1). This correction operator can sometimes be poorly conditioned, so we have incorporated the recommended reduced-rank approach described in [57], [58].

6.4.2 Performance Metrics

Prior to introducing our simulation results, it is instructive to establish performance metrics by which we will evaluate the relative merits of the tested approaches. Performance is evaluated
using aspects of the RFI response to projection operators in the presence of moving RFI based on the operational INR regions depicted in Figure 6.8. Here we present two hypothetical response curves: (1) using the quiescent beamformer weight vector $w_0$, and (2) using modified weights (i.e., $w = Pw_0$) as derived using the algorithm under test. Several vertical region bands identify operational regimes as a function of input INR.

The region labeled “DANGER ZONE!” is the operating range in which interference power is too low to identify the RFI subspace, resulting in poor RFI cancelation and unintended signal suppression. This region can be identified as when the algorithm performs no better than using unmodified weights, which can only happen when the RFI subspace estimate has no relationship to the true underlying RFI subspace.
In the “Identification Region,” interference power levels rise to where the projection begins to cancel the RFI, and performance is primarily limited by signal subspace bias and noise-induced sample estimation error. This region is recognized by the curve’s negative slope, illustrating that as pre-mitigation INR increases, better RFI subspace estimates are obtained so post-mitigation INR decreases.

The “Minimum Region” encompasses interference power levels that lead to minimum post-mitigation RFI power, which is the “sweet spot” for best algorithm performance. This is where the curve has approximately zero slope following the “Identification Region.”

The “Smearing Region” includes interference power levels where the projection operator performance is primarily limited by the motion of the RFI, or when the curve exhibits a positive slope following the “Minimum Region.” The degraded performance in the “Smearing Region” can only be due to RFI motion since the INRs in this region are so large that sample estimation error is negligible. This is also verified if one increases the rank, $Q$, of the moving RFI subspace estimate, which in turn causes the transition point between the “Minimum Region” and “Smearing Region” to move to the right. It is noteworthy that in the “Smearing Region” the positive curve slope matches the slope of the unmitigated curve, meaning that the algorithm provides a fixed level of RFI attenuation, which does not improve with increased pre-mitigation INR.

The key features to examine in the following plots are the transition points between these defined regions and the relative response levels between the tested algorithms within these regions. These will be tabulated for each plot with the corresponding column headings of (1) “ID Pt.,” or the beginning of the RFI identification region, (2) “Min RFI,” or the minimum post-mitigation INR, (3) “Max Depth,” or the maximum null depth in the “Smearing Region,” (4) “Motion Pt.,” or the beginning of the “Smearing Region,” and (5) “SNR Pen.,” or the consequential maximum penalty in SNR from using the algorithm.

6.4.3 Results

Figure 6.9 depicts the performance of the GBT feed without an auxiliary when canceling the GPS satellite BIIR-11 (PRN 19). There are two striking features of the relative performance of the broad null processor compared to the motion-agnostic and motion-cognizant methods: (1) the minimum post-mitigation power for “Broad Null” is the lowest of all the techniques, being
Figure 6.9: The performance of the three approaches on the GBT with $Q = 1$ for the motion-agnostic and motion-cognizant approaches and $Q = 2$ for the broad null approach, an LTI of one second, an STI of 0.01 seconds, and GPS satellite BIIR-11 (PRN 19).

approximately 10 dB lower than the motion-agnostic approach and 17 dB lower than the motion-cognizant algorithm, and (2) in the “Smearing Region,” the achieved maximum null depth is 40 dB deeper than that of the motion-cognizant technique and 64 dB deeper than the motion-agnostic method. In total, the broad null exhibits the best features of both techniques.

Figure 6.10 depicts the response curves when incorporating an additional eigenvector into the motion-agnostic and motion-cognizant projection operators (i.e., $Q = 2$). There is an overall improvement gained when using more eigenvectors with the motion-cognizant approach performing optimally at extremely high interference power levels. However, this improvement is unreliable since the eigenvectors corresponding to the lower-magnitude eigenvalues are heavily biased by sample estimation error. This creates another “Identification Region,” in which the RFI is strong
Figure 6.10: The performance of the three approaches on the GBT with $Q = 2$ for each approach, an LTI of one second, an STI of 0.01 seconds, and GPS satellite BIIR-11 (PRN 19).

enough to be identified by the dominant eigenvector but is too weak for reliable identification of the next-most dominant one.

The transition into the second “Identification Region,” recognized by the pre-mitigation INR value corresponding to the peak value following the “Minimum Region,” begins at approximately 20 dB for the motion-agnostic approach and at approximately 60 dB for the motion-cognizant approach. The broad null approach mitigates the sample estimation error bias in the weaker eigenvectors by correlating rank-one subspace estimates on a per-STI basis. Thus the broad null does not exhibit this second “Identification Region,” making it more reliable for RFI cancelation.

Figure 6.11 shows cancelation performance when using an auxiliary. The largest difference between this and the unassisted case is the starting point of the “Identification Region,” which
Figure 6.11: The performance of the three approaches on the assisted GBT with $Q = 1$ for the motion-agnostic and motion-cognizant approaches and $Q = 2$ for the broad null approach, an LTI of one second, an STI of 0.01 seconds, and GPS satellite BIIR-11 (PRN 19).

occurs at a much lower interference power level, approximately 22 dB lower in the case of the motion-agnostic and broad null methods and 8 dB lower for the motion-cognizant. The only other major difference is the lower minimum RFI level for the motion-agnostic technique. This is a result of the improved INR in the cross-correlation matrix $R$, which allows the RFI subspace to be fully identified before the “Smearing Region” begins.

The downside of the broad null is the use of $Q - 1$ additional degrees of freedom to cancel a single interferer. This results in additional loss in SNR compared to the single-rank projections as shown in Figure 6.12 for the unassisted case. In both the unassisted and auxiliary-assisted cases, the two narrow null approaches suffer from a 0.1 dB loss in SNR while the broad null algorithm suffers a loss of approximately 0.2 dB. The SNR penalty is largely the same in the unassisted case,
Figure 6.12: The effect of the algorithms on the SNR, with $Q = 1$ for the motion-agnostic and motion-cognizant approaches and $Q = 2$ for the broad null.

Table 6.1: Performance on GBT without Auxiliary

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Min RFI</th>
<th>Max Depth</th>
<th>ID Pt.</th>
<th>Motion Pt.</th>
<th>SNR Pen.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion-Agnostic</td>
<td>-40</td>
<td>52</td>
<td>-20</td>
<td>10</td>
<td>0.1</td>
</tr>
<tr>
<td>Motion-Cognizant</td>
<td>-35</td>
<td>92</td>
<td>-20</td>
<td>52</td>
<td>0.1</td>
</tr>
<tr>
<td>Broad Null</td>
<td>-51</td>
<td>116</td>
<td>-20</td>
<td>52</td>
<td>0.22</td>
</tr>
</tbody>
</table>

Note: All values are in dB

which is not depicted here for the sake of brevity. A summary of the technique performance for the GBT in the absence of an auxiliary antenna is shown in Table 6.1, and the performance with an auxiliary is shown in Table 6.2.
Table 6.2: Performance on GBT with Auxiliary

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Min RFI</th>
<th>Max Depth</th>
<th>ID Pt.</th>
<th>Motion Pt.</th>
<th>SNR Pen.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion-Agnostic</td>
<td>-55</td>
<td>52</td>
<td>-52</td>
<td>-11</td>
<td>0.09</td>
</tr>
<tr>
<td>Motion-Cognizant</td>
<td>-35</td>
<td>92</td>
<td>-28</td>
<td>52</td>
<td>0.09</td>
</tr>
<tr>
<td>Broad Null</td>
<td>-55</td>
<td>116</td>
<td>-52</td>
<td>52</td>
<td>0.22</td>
</tr>
</tbody>
</table>

Note: All values are in dB

6.4.4 Canceling Multiple RFI

Figure 6.13 shows unassisted cancelation performance in the presence of two RFI sources, GPS satellites BIIR-11 (PRN 19) and BIIR-13 (PRN 02), where the INR in these plots include contributions from both RFI sources. These results did not include the subspace bias correction step since the correction matrix was so poorly conditioned.

Despite letting $Q = 4$ for each algorithm, the overall cancelation performance is underwhelming, where the maximum attenuation is $\approx 24$ dB for the motion-cognizant approach, 36 dB for motion-agnostic, and 45 dB for the broad null. This decrease in performance is caused largely though the use of the less-dominant eigenvectors, which are heavily biased by noise. Despite these shortcomings, however, the broad null still cancels the multiple RFI more than the alternative algorithms.

As expected, performance improves when utilizing a auxiliary antenna for each RFI source, as shown in Figure 6.14. In this case, $Q = 4$ for the broad null algorithm, and $Q = 2$ for the motion-agnostic and motion-cognizant approaches since the maximum number of available degrees of freedom is equal to the number of auxiliary antennas. Interestingly, the motion-cognizant algorithm achieves a slightly higher maximum attenuation level of approximately 26.5 dB, and the motion-agnostic approach is degraded with a maximum attenuation of approximately 34.5 dB. The broad null algorithm, however, achieves the same maximum attenuation level as in the unassisted case above and still demonstrates superior overall canceling capability compared to the other tested approaches.

6.4.5 STI Length Considerations

The number of STI windows per LTI window (i.e., $L$) is another critical parameter to be considered in the evaluation of the broad null algorithm. The effect of two extreme STI lengths for
a single-second LTI is shown in Figure 6.15. For reference, we have also included results using the motion-cognizant method. Surprisingly, the broad null algorithm performance appears largely unaffected by the selection of STI length. Therefore, we conclude that one can reliably use the broad null method with very large STI lengths, with the maximum being $1/Q$th of the LTI length since the number of STI windows $L$ must be greater than or equal to $Q$ in order to extract enough eigenvectors from $\hat{R}_U$ in (6.29).

The conclusion that the broad null approach performs well with longer STI lengths is a striking one since it indicates that one can reduce computational complexity by increasing the STI length to its maximum without sacrificing cancelation performance. By setting $L = Q$, the output
Figure 6.14: The performance of the three approaches on the assisted GBT with $Q = 2$ for the motion-cognizant and agnostic algorithms and $Q = 4$ for the broad null, an LTI of one second, an STI of 0.01 seconds, and two RFI sources: GPS satellites BIIR-11 (PRN 19) and BIIR-13 (PRN 02).

correlator data rate when assisted becomes

$$d = K_c \left( \frac{OM}{T_{LTI}} + \frac{M(M - 1)}{2T_{LTI}} \right) b,$$

(6.43)

and the unassisted correlator data rate becomes

$$d = K_c \left( \frac{OM(M + 1)}{2T_{LTI}} \right) b.$$  

(6.44)

The resulting data rates for the FLAG and ASKAP instruments are summarized in Table 6.3.
Figure 6.15: The performance of the broad null and motion-agnostic approach with $Q = 2$ and varying STI lengths. The LTI length is one second, and the interfering source is GPS satellite BIIR-11 (PRN 19).

Table 6.3: Data Rates for Various Approaches

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>FLAG</th>
<th>FLAG+aux</th>
<th>ASKAP</th>
<th>ASKAP+aux</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion-Agnostic</td>
<td>0.023</td>
<td>0.025</td>
<td>0.073</td>
<td>0.074</td>
</tr>
<tr>
<td>Motion-Cognizant, $L = 100$</td>
<td>2.371</td>
<td>2.496</td>
<td>7.727</td>
<td>7.354</td>
</tr>
<tr>
<td>Broad Null, $L = 100$</td>
<td>2.371</td>
<td>1.441</td>
<td>7.727</td>
<td>0.149</td>
</tr>
<tr>
<td>Broad Null, $L = 2$</td>
<td>0.047</td>
<td>0.025</td>
<td>0.146</td>
<td>0.146</td>
</tr>
</tbody>
</table>

Note: All values are in gigabits-per-second (Gbps)
One particular item to highlight from Table 6.3 is that the data rates for auxiliary assisted case when using the broad null, $L = 2$ is only marginally higher (not apparent due to insufficient significant digits) than when using the motion-agnostic approach. Therefore, the maximum-STI-length broad null approach is approximately as computationally expensive in terms of correlator output rate as the motion-agnostic approach. Thus the broad null algorithm is also a computationally affordable approach to projection-based RFI mitigation.

### 6.4.6 LTI Length Considerations

The length of each LTI window also plays a role in the performance of the broad null approach to RFI mitigation, which is shown in Figure 6.16 for three tested LTI lengths. Note that for comparison, we have superimposed two curves representing results for the motion-cognizant approach using two different STI lengths.

From these results, we see that when the LTI length becomes larger, the broad null undergoes a left shift of the transition point to the “Smearing Region,” indicating reduced motion tracking capability. However, in contrast, the minimum value in the “Minimum Region” decreases with increased LTI length, indicating improved cancelation capability for moderately strong RFI. As such, one can specify a required level of RFI motion mitigation that can be achieved using the motion-cognizant approach and find the LTI length that will meet that requirement and also provide the the deepest broad null. In this case, if the specified motion cancelation requirement is satisfied by the “Motion Cognizant, STI = 0.5” curve in Figure 6.16, then the broad null algorithm with an LTI just less than 5s will achieve the same motion mitigation with an overall deeper null.

### 6.5 Conclusion

We have explored two projection-based approaches to moving-RFI mitigation when SNR is less than unity and the antenna array is not calibrated in the angular region of the RFI, and introduced a new broad-null forming algorithm. This broad null approach was shown to be superior across almost all observed metrics except for a marginal decrease in SNR. The proposed algorithm increases the maximum null depth, decreases the residual RFI power, and mitigates moving RFI
more effectively than its counterparts. It also greatly reduces the overall data load of a real-time correlator.

Lastly, the simulation parameters used herein match the exact specifications and current capabilities of the FLAG radio astronomical PAF system, enabling immediate adoption of the broad null method. A single ASKAP PAF system is currently only able to output single-second correlation matrices with a 50% duty cycle, but if it adopted the proposed auxiliary-assisted architecture, there is little more that must be implemented in order to utilize broad nulls.
CHAPTER 7. CONCLUSIONS AND FUTURE WORK

In this work, we have enhanced the capabilities of antenna array systems used in radio astronomy and expanded the foundational detection, estimation, and interference mitigation theory surrounding them. This was done by deriving maximum-likelihood (ML) transient detectors and critical array parameter estimators, performing a Cramér Rao bound (CRB) analysis for astronomical transient parameters, developing a wideband, multi-purpose phased array feed (PAF) system, and created an improved and computationally affordable moving interference filtering algorithm.

The PAF system that was developed, the Focal L-Band Array for the Green Bank Telescope (FLAG), represents the most recent efforts in world-class PAF engineering, having dramatically decreased HI observation times by nearly a factor of four. The FLAG digital back end, having been implemented in software on graphics processing unit (GPU) cards, will be an invaluable tool for the testing and development of real-time advanced interference mitigation techniques and other novel signal processing algorithms. This instrument will hopefully also be used to perform high-profile science observations and kick-start the development of even more advanced array systems in the future.

Ultimately, this work represents a small step in the innovation of array receiver systems in radio astronomy. In the future, these findings need to be put into practice and refined. The ML transient detectors presented herein must be simulated and compared against traditional approaches. The CRB analysis must be expanded to accommodate more realistic signal models that incorporate non-idealities such as phase jitter (when a pulse profile varies in time). Existing pulsar timing algorithms (parameter estimators) should also be simulated and evaluated against the CRB. The FLAG instrument should also be fully integrated into the Green Bank mainframe as a common-user system and upgraded with a concurrent beamformer/correlator module. The theory behind the broad null algorithm should also be adapted for interference mitigation in arrays with large baselines and applied in radar chirp spectrum estimation techniques.
Existing instruments such as FLAG and the VLA must add real-time broad-null-forming RFI-canceling implementations. The algorithm must then undergo some significant real-data testing and then be moved into operational status and become widely used by astronomers. Without significant results recovering astronomical data in the presence of RFI, significant bands of interest will continue to corrupted thus limiting the advance of science.

Equipped with both a flexible PAF receiver system such as FLAG and the theoretical developments in this work and those to come, astronomers will begin to detect more exotic transients like fast radio bursts (FRBs) and penetrate the ever-growing veil of man-made interference to continue uncovering the many mysteries of the universe.
REFERENCES


142


144


APPENDIX A. DERIVATIONS OF JACOBIANS

A.1 Jacobian for Single Antenna Model

Recall the parameter vector as
\[ \alpha = [D, t_0, T, \sigma^2_s, \cdots, \sigma^2_{s,K}, \sigma^2_z, \cdots, \sigma^2_{z,K}]^T, \]  
(A.1)
\[ = [D, t_0, T, \sigma^T_s \sigma^T_z]^T, \]  
(A.2)
where \( \sigma_s = [\sigma^2_s, \cdots, \sigma^2_{s,K}]^T \), and \( \sigma_z \) is defined similarly. Given the modified definition of the Jacobian,
\[ \tilde{J} = \frac{\partial \text{diag}\{R_V\}}{\partial \alpha^T}, \]  
(A.3)
where the diagonal entries of \( R_V \) are \( \sigma^2_{v,k,m} \), we can restructure \( \tilde{J} \) such that
\[ \tilde{J} = [j_{0,0}, \cdots, j_{K,M}]^T, \]  
(A.4)
where
\[ j_{k,m} = \frac{\partial \sigma^2_{v,k,m}}{\partial \alpha} \]  
(A.5)
\[ = [j_{D,k,m}, j_{t_0,k,m}, j_{T,k,m}, j_{\sigma_s,k,m}, j_{\sigma_z,k,m}]^T. \]  
(A.6)
These quantities are given by

\[
\begin{align*}
\dot{j}_{D,k,m} &= \frac{\partial \sigma_{\nu,k,m}^2}{\partial D} = 2h_k(D,m,t_0,T) \frac{\partial h_k(D,m,t_0,T)}{\partial D} \sigma_{s,k}^2 a(\theta,\phi) \\
\dot{j}_{t_0,k,m} &= \frac{\partial \sigma_{\nu,k,m}^2}{\partial t_0} = 2h_k(D,m,t_0,T) \frac{\partial h_k(D,m,t_0,T)}{\partial t_0} \sigma_{s,k}^2 a(\theta,\phi) \\
\dot{j}_{T,k,m} &= \frac{\partial \sigma_{\nu,k,m}^2}{\partial T} = 2h_k(D,m,t_0,T) \frac{\partial h_k(D,m,t_0,T)}{\partial T} \sigma_{s,k}^2 a(\theta,\phi) \\
\dot{j}_{\sigma,\nu,k,m}^T &= \frac{\partial \sigma_{\nu,k,m}^2}{\partial \sigma_{z}^T} = h_k^2(D,m,t_0,T) e_k^T a(\theta,\phi) \\
\dot{j}_{\sigma,\nu,k,m} &= \frac{\partial \sigma_{\nu,k,m}^2}{\partial \sigma_{z}} = e_k^T,
\end{align*}
\]

where \(e_k\) is a \(K \times 1\) column vector with a one in the \(k\)th entry and zeros elsewhere. For the pulse model in (4.8),

\[
\begin{align*}
\dot{j}_{D,k,m} &= -\frac{1}{T} \exp \left\{ \frac{-(mT_s-t_0+2T+m_D(k))^2}{2T} \right\} a_k(\theta,\phi) \times \\
&\quad (mT_s-t_0+2T+m_D(k))(c(f_0^{-2} - f_k^{-2})) \sigma_{s,k}^2, \\
\dot{j}_{t_0,k,m} &= \frac{1}{T} \exp \left\{ \frac{-(mT_s-t_0+2T+m_D(k))^2}{2T} \right\} a(\theta,\phi) \times \\
&\quad (mT_s-t_0+2T+m_D(k)) \sigma_{s,k}^2, \\
\dot{j}_{T,k,m} &= \frac{1}{2T^2} \exp \left\{ \frac{-(mT_s-t_0+2T+m_D(k))^2}{2T} \right\} a(\theta,\phi) \times \\
&\quad [(mT_s-t_0+2T+m_D(k))^2 \times \\
&\quad -4T(mT_s-t_0+2T+m_D(k)) \sigma_{s,k}^2 a(\theta,\phi)], \\
\dot{j}_{\sigma,\nu,k,m} &= \exp \left\{ \frac{-(mT_s-t_0+2T+m_D(k))^2}{2T} \right\} a(\theta,\phi) e_k, \\
\dot{j}_{\sigma,\nu,k,m} &= e_k.
\end{align*}
\]

A.2 Jacobian for Array Antenna Model

Recall that the parameter for an array antenna is given by

\[
\alpha = [D,t_0,T,\sigma_{s}^T,\theta,\phi,\text{vec}\{R_{z,0}\}^T,\ldots,\text{vec}\{R_{z,K-1}\}^T].
\]
Therefore,

\[
\frac{\partial}{\partial D} {r}^{(p,q)}_{v,k,m} = 2h_k(D,m,t_0,T)\frac{\partial h_k(D,m,t_0,T)}{\partial D}\sigma_{s,k}^2 a_k^{(p)}(\theta,\phi) a^*_k(\theta,\phi),
\]

(A.18)

\[
\frac{\partial}{\partial t_0} {r}^{(p,q)}_{v,k,m} = 2h_k(D,m,t_0,T)\frac{\partial h_k(D,m,t_0,T)}{\partial t_0}\sigma_{s,k}^2 a_k^{(p)}(\theta,\phi) a^*_k(\theta,\phi),
\]

(A.19)

\[
\frac{\partial}{\partial T} {r}^{(p,q)}_{v,k,m} = 2h_k(D,m,t_0,T)\frac{\partial h_k(D,m,t_0,T)}{\partial T}\sigma_{s,k}^2 a_k^{(p)}(\theta,\phi) a^*_k(\theta,\phi),
\]

(A.20)

\[
\frac{\partial}{\partial \sigma_T} {r}^{(p,q)}_{v,k,m} = h_k^2(D,m,t_0,T)\sigma_{s,k}^2 e_k^T a_k^{(p)}(\theta,\phi) a^*_k(\theta,\phi),
\]

(A.21)

\[
\frac{\partial}{\partial \theta} {r}^{(p,q)}_{v,k,m} = h_k^2(D,m,t_0,T)\sigma_{s,k}^2 \frac{\partial a_k^{(p)}(\theta,\phi) a^*_k(\theta,\phi)}{\partial \theta},
\]

(A.22)

\[
\frac{\partial}{\partial \phi} {r}^{(p,q)}_{v,k,m} = h_k^2(D,m,t_0,T)\sigma_{s,k}^2 \frac{\partial a_k^{(p)}(\theta,\phi) a^*_k(\theta,\phi)}{\partial \phi}.
\]

(A.23)

The noise covariance entries are given by

\[
\frac{\partial}{\partial \text{vec} \{ R_{z,k} \}} {r}^{(p,q)}_{v,k,m} = \text{vec} \left\{ \frac{\partial}{\partial \text{vec} \{ R_{z,k} \}} R_{v,k,m}^{(p,q)} \right\}^T,
\]

(A.24)

\[
\frac{\partial}{\partial \text{vec} \{ R_{z,j\neq k} \}} {r}^{(p,q)}_{v,k,m} = 0_j^T,
\]

(A.25)
where $E_{p,q}$ is a $J \times J$ matrix of zeros with a one in the $p$th row and $q$th column. Therefore, using the pulse shape model from (4.8),

$$
\frac{\partial}{\partial D} r_{v,k,m}^{(p,q)} = -\frac{1}{T} \exp \left\{ \frac{-(mT_s - t_0 + 2T + mD(k))^2}{2T} \right\} \times \\
(mT_s - t_0 + 2T + mD(k)) \times \\
(c(f_0^{-1} - f_k^{-2})) \sigma^2_{s,k} a_k^{(p)}(\theta, \phi) a_k^{\ast(q)}(\theta, \phi),
$$

(A.26)

$$
\frac{\partial}{\partial t_0} r_{v,k,m}^{(p,q)} = \frac{1}{T^2} \exp \left\{ \frac{-(mT_s - t_0 + 2T + mD(k))^2}{2T} \right\} \times \\
(mT_s - t_0 + 2T + mD(k)) \times \\
\sigma^2_{s,k} a_k^{(p)}(\theta, \phi) a_k^{\ast(q)}(\theta, \phi),
$$

(A.27)

$$
\frac{\partial}{\partial T} r_{v,k,m}^{(p,q)} = \frac{1}{T^2} \exp \left\{ \frac{-(mT_s - t_0 + 2T + mD(k))^2}{2T} \right\} \times \\
[(mT_s - t_0 + 2T + mD(k))^2 \times \\
-4T(mT_s - t_0 + 2T + mD(k))] \times \\
\sigma^2_{s,k} a_k^{(p)}(\theta, \phi) a_k^{\ast(q)}(\theta, \phi),
$$

(A.28)

$$
\frac{\partial}{\partial \theta} r_{v,k,m}^{(p,q)} = \exp \left\{ \frac{-(mT_s - t_0 + 2T + mD(k))^2}{2T} \right\} \times \\
\sigma^2_{s,k} \frac{\partial a_k^{(p)}(\theta, \phi)}{\partial \theta},
$$

(A.29)

$$
\frac{\partial}{\partial \phi} r_{v,k,m}^{(p,q)} = \exp \left\{ \frac{-(mT_s - t_0 + 2T + mD(k))^2}{2T} \right\} \times \\
\sigma^2_{s,k} \frac{\partial a_k^{(p)}(\theta, \phi) a_k^{\ast(q)}(\theta, \phi)}{\partial \phi},
$$

(A.30)

$$
\frac{\partial}{\partial \sigma^T s} r_{v,k,m}^{(p,q)} = \exp \left\{ \frac{-(mT_s - t_0 + 2T + mD(k))^2}{2T} \right\} \times \\
\mathbf{e}^T_k a_k^{(p)}(\theta, \phi) a_k^{\ast(q)}(\theta, \phi),
$$

(A.31)

and the noise covariance terms are given by

$$
\frac{\partial}{\partial \text{vec} \{ \mathbf{R}_{s,k} \}^T} r_{v,k,m}^{(p,q)} = \text{vec} \{ E_{p,q} \}^T.
$$

(A.32)

Since the array manifold $a_k(\theta)$ is not easily modeled due to real-world complications such as imperfections in a reflector, antenna mutual coupling, and scattering effects, we will use a numerical
approach to computing $a_k(\theta)$ and $\frac{\partial a_k^{(p)}(\theta) a_k^{(q)}(\theta)}{\partial \theta}$ such that

$$\frac{\partial a_k^{(p)}(\theta)}{\partial \theta} \approx \frac{a_k^{(p)}(\theta + \Delta \theta) a_k^{(q)}(\theta + \Delta \theta) - a_k^{(p)}(\theta - \Delta \theta) a_k^{(q)}(\theta - \Delta \theta)}{2\Delta \theta}. \quad (A.33)$$

To ensure a good numerical approximation, we will let $2\Delta \theta$ be no more than 1/10th of a beam width.
APPENDIX B. FLAG DEVELOPMENT DETAILS

B.1 xGPU Details

The library is compiled to be C compatible so that it can be linked to HASHPIPE. The xgpu.so library supports the coarse-channel correlator; the xgpu_pfb.so library supports the fine-channel correlator; and the xgpu_frb.so library supports the rapid-dump correlator.

B.2 HASHPIPE Plugins

When HASHPIPE is run on the command line as-is, it will result in nothing since it requires a plugin that specifies the application to be run. In a sense, a HASHPIPE plugin is analogous to a program run on any other conventional RTOS, but is distinct in that a HASHPIPE program is in the form of a shared library. Thus, linking a shared library to HASHPIPE at run-time is like “plugging in” the software; hence the “plugin” terminology.

A HASHPIPE plugin is basically a shared object that defines various threads and shared memory buffer parameters. Each thread is encapsulated in a single .c file that contains a constructor function that is called by hashpipe when its thread is created. An example constructor is show below.

```c
static __attribute__((constructor)) void ctor() {
    register_hashpipe_thread(&thread_desc);
}
```

The argument thread_desc is a hashpipe_thread_desc_t struct defined in the thread’s .c file as follows:

```c
static hashpipe_thread_desc_t thread_desc = {
    name: "thread_name", // Name of the thread
    skey: "KEYNAME", // Thread status memory keyword
    init: init_func, // Name of function to run when initializing the thread
```
After compilation, a shared library object (.la) is generated that represents this pipeline with its various threads and semaphores. When the library is placed in a directory found in the LD_LIBRARY_PATH environment variable, it can be executed using the following at a terminal:

```
$ hashpipe -p <library_name>
```

where `<library_name>` is the shared library file name without its .la extension. To run multiple simultaneous instances, one can use the `-I <instance_num>` flag. Shared memory status values can be set using the `-o <KEY>=<value>` flag. Lastly, core affinities for the various threads are set using the `-c <core_num> <thread_name>` flag. For example,

```
$ hashpipe -p flag_beamformer -I 0 -o XID=0 -o BINDHOST=10.10.10.1 -o GPUDEV=0
   -c 0 flag_net_thread -c 1 flag_transpose_thread -c 2 flag_beamformer_thread
```

A simple “Hello World” HASHPIPE plug-in can be found in Appendix C.

B.3 HASHPIPE Threads

For each specified configuration, there is an associated HASHPIPE plugin, named as follows:

1. `flag_bx.la` → Coarse Channel, Rapid Dump, Reduced Bandwidth Correlator/Coarse Beamformer

2. `flag_bfx.la` → Fine Channel Correlator/Coarse Beamformer

There are also other reduced capability configurations that are available for simple tests or diagnostics, which are named:

1. `flag_x.la` → Coarse PFB Correlator

2. `flag_x_frb.la` → Coarse PFB Rapid Dump, Reduced Bandwidth Correlator
3. `flag_b.la` → Coarse Beamformer

4. `flag_f.la` → Fine PFB

5. `flag_fx.la` → Fine PFB Correlator

These HASHPIPE plugins use many threads, but they all align into roughly the same structure, shown in Figure B.1. In essence, the first thread must always be a network sniffing thread, which listens on the network for packets from the ROACH boards. The second thread must always be a transpose thread, which reformats the data so that all time samples are co-located in memory (details later). The next thread should be selected based on desired operational mode, and the final thread is a save-to-disk thread for debugging purposes. If the FITS writers are being used, the fourth thread should be omitted.

We here enumerate the threads used in each configuration:

**The flag_bx.la Mode**

1. `flag_net_thread`

2. `flag_bx_transpose_thread`

3. `flag_bx_thread`

4. (For Debugging) `flag_bx_save_thread`

**The flag_bfx.la Mode**
1. flag.net_thread
2. flag.transpose_thread
3. flag.bfx_thread
4. (For Debugging) flag.bfx.corsave_thread

The flag_x.la Mode

1. flag.net_thread
2. flag.transpose_thread
3. flag.correlator_thread
4. (For Debugging) flag.corsave_thread

The flag_x_frb.la Mode

1. flag.net_thread
2. flag.frb.transpose_thread
3. flag.frb.correlator_thread
4. (For Debugging) flag.frb.corsave_thread

The flag_b.la Mode

1. flag.net_thread
2. flag.transpose_thread
3. flag.beamform_thread
4. (For Debugging) flag.beamsave_thread

The flag_f.la Mode

1. flag.net_thread
2. flag_transpose_thread
3. flag_pfb_thread
4. (For Debugging) flag_pfbsave_thread

The flag_fx.la Mode

1. flag_net_thread
2. flag_transpose_thread
3. flag_pfb_thread
4. flag_pfb_correlator_thread
5. (For Debugging) flag_pfb_corsave_thread

This section will review each thread and its functionality.

B.3.1 flag_net_thread

The flag_net_thread listens for incoming UDP packets from the ROACH boards and stores them in a buffer for further processing. It also handles incoming messages from the Player and subsequently commands the remaining threads in the process.

The internal processing structure is a state machine, which is depicted in Figure B.2. Here, the thread waits in an IDLE state until the Player issues a “start” command through stdin. Once the “start” command is issued, the thread enters the ACQUIRE state in which it begins to listen for ROACH packets and stores any that it finds. When it stores a block of data whose starting frame counter (mcnt) is greater than or equal to the specified last frame counter (i.e., scan length), or when the Player issues a “stop” command, the thread transitions into a CLEANUP state. In CLEANUP, the thread issues the same command through shared memory to the other threads and reinitializes the input buffer and all relevant counters to prepare for a new scan. Then the thread returns to IDLE to wait for a new “start” command.

Since the full-data-rate system sends packets through a network switch, out-of-order packets are very common. Consequently this thread incorporates a large output buffer with many
Figure B.2: The state machine diagram for flag_net_thread.

Figure B.3: The data coming out of the flag_net_thread are ordered such each block contains 200 frames, which each contain eight ROACH packets. Each packet contains 20 time samples of 25 frequency channels across eight antenna elements. Blocks, where each block contains 200 packets worth of data, which translates to 4000 complex time samples of 25 frequency channels across 64 antenna elements. The data are ordered in the buffer according to the diagram shown in Figure B.3.

B.3.2 flag_transpose_thread

Unlike from flag_net_thread, this thread uses a much simpler state machine, depicted in Figure B.4. Here, the thread resides in an ACQUIRE state in which it waits for the input buffer to become populated and reformats the data so as to aggregate the time samples together as depicted in Figure B.5. When the flag_net_thread goes into a CLEANUP state, this thread also enters a CLEANUP state in which it resets all relevant counters.
Figure B.4: The state machine diagram for every thread except flag_net_thread.

Figure B.5: The data format coming out of the flag_transpose_thread. Each block contains 4000 time samples of 25 frequency channels across 64 antenna elements.

### B.3.3 flag_pfb_transpose_thread

As with the flag_transpose_thread, this thread uses the same state machine depicted in Figure B.4. The operation of the this thread is the same as that of flag_transpose_thread except that the output consists of only five of the total 25 frequency channels. One can control which five frequency channels are processed by writing a value to the shared memory status keyword CHANSEL, where a value of $k$, $0 \leq k \leq 4$ yields channels $5k$ through $5k + 4$. These outputs are then marked as ready for the flag_frb_correlator_thread to process.

### B.3.4 flag_frb_transpose_thread

As with the flag_transpose_thread, this thread uses the same state machine depicted in Figure B.4. The operation of the this thread is the same as its non-FRB counterpart above except that the output consists of only five of the total 25 frequency channels and is split into 200 blocks per one input block. This creates output blocks that span 40 time samples, or approximately 0.13
milliseconds. One can control which five frequency channels are processed by writing a value to the shared memory status keyword CHANSEL, where a value of \( k \), \( 0 \leq k \leq 4 \) yields channels \( 5k \) through \( 5k + 4 \). These outputs are then marked as ready for the flag.frb.correlator.thread to process.

B.3.5 flag.bx.transpose_thread

As with the other transpose threads previously described, this thread uses the same state machine depicted in Figure B.4. This thread sources two output buffers, the first for the real-time beamformer operation and the second for the fast-dump, reduced-bandwidth correlator managed by the flag.bx.thread. The beamformer’s buffer is organized as described in Section B.3.2, and the correlator’s buffer is organized as described in Section B.3.4.

B.3.6 flag.pfb_thread

This thread accesses the 8-bit data outputted by the flag.pfb.transpose_thread and calls the PFB GPU codes to further channelize the data. The user can specify before run-time the filter length and window type by using other utilities found with the PFB library. Details about the PFB library are beyond the scope of this work as they were developed by Mitchell Burnett.

B.3.7 flag.correlator_thread

This thread accesses the 8-bit data outputted by the flag.transpose_thread and calls the xGPU process that will correlate the data. The user can specify an integration length by writing the desired number of seconds to the REQSTI value of the status shared memory. The integration length must be a multiple of 4000 time samples, or approximately 13 ms, so the code rounds up to the nearest multiple of 4000 time samples. As with the flag.bx.transpose_thread, this thread uses the state machine depicted in Figure B.4. The correlation matrix format has the same structure depicted in Figure B.6.
Figure B.6: The data format coming out of the flag_correlator_thread.

B.3.8 flag_frb_correlator_thread

This thread accesses a single block produced by the flag_frb_transpose_thread and calls the xGPU process that will correlate the data. In this thread, the xGPU process is linked to the xgpu_frb.so library, which supports only five frequency channels and processes blocks of 40 time samples. As with the flag_correlator_thread, the output data format is depicted in Figure B.6.

B.3.9 flag_pfb_correlator_thread

This thread computes spatial correlation matrices for 160 fine frequency channels provided by the flag_pfb_thread. Here, the xGPU process is linked to the xgpu_pfb.so library, which expects floating point complex values for 160 frequency channels in 4000-time-sample blocks. As with the other correlator threads, the output data format is given by the format in Figure B.6.

B.3.10 flag_beamform_thread

This thread accesses the data output by the flag_transpose_thread and processes it using the real-time beamformer GPU kernel. To this end, it manages the beamformer weights and
populates shared memory with relevant metadata, which is itemized in Table B.1. The thread also uses the state machine from Figure B.4. The output data product format is a three-dimensional array with dimensions of $25 \times 4 \times 7$ (frequency channel $\times$ polarization $\times$ beam index), where beam index is the fastest changing dimension followed by polarization ($XX \rightarrow YY \rightarrow XY$).

### B.3.11 flag\_bx\_thread

This thread creates two sub-threads of its own: the first implements the real-time fast-dump correlator as used in the flag\_frb\_correlator\_thread, and the second implements the real-time beamformer as used in the flag\_beamform\_thread. To accommodate the different output data products, two data buffers are created, the first being the default associated with the thread and the second created manually before the two sub-threads are created. The two sub-threads independently use the state machine from Figure B.4.

### B.3.12 flag\_bfx\_thread

This thread is not yet implemented, but is meant to take data from the flag\_transpose\_thread and apply the real-time beamformer, fine-channelization PFB, and correlator in parallel on the data. This single thread approach may not be the ideal approach to achieve a concurrent operational mode, and is left to those following to evaluate and implement.

### B.3.13 flag\_corsave\_thread

This optional thread must follow the flag\_correlator\_thread. It saves the correlator outputs defined in Figure B.6 across 20 frequency channels into a raw text format, with each
new line containing a single floating-point sample. Every other sample is real, and the remaining samples are imaginary.

**B.3.14 flag frb_corsave_thread**

This optional thread must follow the `flag_frb_correlator_thread`. It saves the short-time dumps of the correlator formatted according to Figure B.6 across five fine frequency channels into a raw text format, with each new line containing a single floating-point sample. Every other sample is real, and the remaining samples are imaginary.

**B.3.15 flag pfbsave_thread**

This optional thread must follow the `flag_pfb_thread`. It saves the PFB channels into a raw text format, with each new line containing a single floating-point sample. Every other sample is real, and the remaining samples are imaginary. The exact format of the data is beyond the scope of this work as it was developed by Mitchell Burnett.

**B.3.16 flag pfb_corsave_thread**

This optional thread must follow the `flag_pfb_correlator_thread`. It saves the correlator outputs defined in Figure B.6 across 160 frequency channels into a raw text format, with each new line containing a single floating-point sample. Every other sample is real, and the remaining samples are imaginary.

**B.3.17 flag beamsave_thread**

This optional thread must follow the `flag_beamform_thread`. It saves the beamformer outputs defined in Figure B.6 into a binary file consisting of only float pairs, each representing a complex sample.
B.3.18  flag\_bx\_save\_thread

This optional thread must follow the flag\_bx\_thread. It creates two sub-threads that emulate the behavior of the flag\_frb\_corsave\_thread and flag\_beamsave\_thread codes.

B.3.19  flag\_bfx\_save\_thread

This optional thread must follow the flag\_bfx\_thread. It creates two sub-threads that emulate the behavior of the flag\_pfb\_corsave\_thread and flag\_beamsave\_thread codes.

B.4  Dealer/Player Implementation Details

The Dealer/Player system is a Python-based code suite that manages ROACH boards and HPC processes and provides mechanisms to issue user commands and control shared memory. Detailed documentation for the Dealer/Player system can be found at [125]. In summary, a “Dealer” acts as a server, issuing commands to its “Players” or clients via zero-MQ socket protocols.

The FLAG back end system has a single Dealer that communicates with 20 Players that each manage a single instance of our HASHPIPE code and FITS writer. A single Player is shown in Figure B.7. Here the Dealer issues commands to the Player, which in turn translates that command into messages that are then fed into the stdin of the HASHPIPE and FITS writer processes.

The Dealer/Player interface is controlled by a configuration file named dibas.conf, which outlines the parameters for each Player instance, known as BANKs, and the various operational modes that the Player can turn on. An example of a BANK configuration is shown below:

```
[BANKA]
# HPC / Player host & port
hphost = flag4
player_port = 6677
# ROACH Control:
has_roach = true
katcp_ip = byur2
katcp_port = 7147
# Data flow
data_source_host = byur2
data_source_port = 60000
data_destination_host = 10.10.1.13
data_destination_port = 60000
```
Figure B.7: Block diagram depicting the control interfaces between the Dealer, Player, HASHPIPE codes, FITS writer, and shared memory.

# Synthesizer:
synth = none
# I'm pretty sure these don't matter if katcp is set as the synthesizer
synth_port = /dev/ttyS1
synth_ref = external
synth_ref_freq = 10000000
synth_vco_range = 2200, 4400
synth_rf_level = 5
synth_options = 0,0,1,0
# I2C (nonsense for FLAG)
filter_bandwidth_bits = 450, 0x00, 1450, 0x08, 1900, 0x18
# FLAG-specific parameters
xid = 12
instance = 0
gpudev = 0
cpus = 0, 1, 2, 3

An example configuration for a mode named FLAG_HICORR_MODE is shown below:

[FLAG_HICORR_MODE]
# These values get loaded directly into status shared memory
shmkeys = BACKEND, MODENAME
A list of all the modes that the FLAG back end supports are shown in Table B.2.

### B.5 Scan Overlord

The Scan Overlord is a Python script that monitors the state of the Green Bank Telescope scan coordinator and issues commands to the Dealer. Communication with the scan coordinator is accomplished by creating a map of desired values to observe and the respective call-
back functions that should be called. For example, the key to get the scan coordinator state is ScanCoordinator.ScanCoordinator:P:state. We can then assign a callback function to run whenever that scan coordinator state changes, so that

```
keys = {"ScanCoordinator.ScanCoordinator:P:state": state_callback},
```

where the method state_callback is called when the state changes.

Before proceeding, the following include statements will be needed:

```python
import dealer
from ZMQJSONProxy import ZMQJSONProxyException
import zmq
import sys
from PBDataDescriptor_pb2 import PBDataField
from DataStreamUtils import get_service_endpoints
```

To connect to the scan coordinator, one must being listening on a socket, which is done as follows:

```
req_url = "tcp://gbtdata.gbt.nrao:5559"
ctx = zmq.Context()
subscriber = ctx.socket(zmq.SUB)
```

This creates a ZMQ subscriber socket listening to the ScanCoordinator. One can then receive all messages from the ScanCoordinator by setting the socket filter option to an empty string.

```
subscriber.setsockopt(zmq.SUBSCRIBE, '')
```

In our case, we are interested in only some messages, which are specified by the key values from the keys dictionary, and the socket filters can be incorporated as follows:

```
for key in keys:
    major, minor = key.split(':')[:-1].split('.')
    sub_url, _, _ = get_service_endpoints(ctx, req_url, major, minor, 0)
    subscriber.connect(sub_url)
    subscriber.setsockopt(zmq.SUBSCRIBE, key)
```

Then one must start a main loop that constantly listens for messages from the scan coordinator, as follows:
while (auto_set):
    key, payload = subscriber.recv_multipart()
    df = PBDataField() # Create a message parser/decoder
    df.ParseFromString(payload) # Parse the string message into a struct
    f = keys[key] # Get the callback function name for the message
    f(df) # Call the callback function

Note that PBDataField is a Google Developer library, where “PB” stands for “Protocol Buffer.” This library contains a string-parsing code that converts the received message into a struct.

This struct is the single argument of the respective callback function. For example, the struct received by state_callback contains a list of structures called val_struct, whose first entry contains a list of strings called val_string. Thus, one can print the current state using the following callback function:

```python
def state_callback(p):
    print p.val_struct[0].val_string[0]
```

Currently, there are only four callback functions are in place, resulting in the key-callback map of

```python
keys = {
    "ScanCoordinator.ScanCoordinator:P:state": state_callback,
    "ScanCoordinator.ScanCoordinator:P:startTime": start_time_callback,
    "ScanCoordinator.ScanCoordinator:P:scanLength": scan_len_callback,
    "ScanCoordinator.ScanCoordinator:P:projectId": project_id_callback
}
```

The other callback functions can access the data as shown below:

```python
def start_time_callback(p):
    num_sec = p.val_struct[0].val_struct[0].val_double[0]

def scan_len_callback(p):
    scan_length = p.val_struct[0].val_struct[0].val_double[0]

def project_id_callback(p):
    project_id = p.val_struct[0].val_string[0]
```

### B.6 MATLAB Post-Processing Codes

A suite of post-processing codes for the FLAG receiver were developed and are saved on a remote GIT repository located at

[https://github.com/rallenblack/matFlag](https://github.com/rallenblack/matFlag)

As of the latest commit (7e4aeel) at the time of this writing, there were six sub-directories:
Each subsection that follows will explore the codes in each sub-directory mentioned above.

### B.6.1 kernel

The codes located in the `kernel` sub-directory are functions and scripts that are called by scripts in the other sub-directories. These include scripts that will (1) open FITS files and parse the data into a MATLAB-friendly format, (2) extract antenna position information from ancillary FITS files, and (3) tabulate scan metadata from scan logs.

#### extract_covariances

This function opens a FITS file containing correlations from the coarse-channel correlator and reconstructs the correlation matrix. It then returns the restructured matrix, the corresponding DMJD, XID, and FITS header information. The function declaration is as follows:

```matlab
function [ R, dmjd, xid, info ] = extract_covariances( fits_filename )
```

A note to users and developers: this function is not fast since file I/O in MATLAB is slow. It is possible that some micro optimization could speed this method up.

#### extract_pfb_covariances

This function performs the same operations as in `extract_covariances` except that it is expecting correlations from the fine-channel correlator. The function declaration is as follows:

```matlab
function [ R, dmjd, xid, info ] = extract_pfb_covariances( fits_filename )
```
**extract_bf_output**

This function opens a FITS file containing beamformed powers from the real-time beamformer and reconstructs the data into a matrix with dimensions (Beams × Polarization × Frequency × Time). The function is a little quirky since the default MATLAB FITS file reader crashes with large file sizes, and so care must be taken when using this. Details about this method are beyond the scope of this work since it was developed by Mark Ruzindana. The function declaration is as follows:

```matlab
function [ B, xid ] = extract_bf_output( fits_filename )
```

**aggregate_banks**

This function collects the correlations from every BANK and the corresponding antenna positions, and aggregates them. The correlations are averaged in time based on user input, and the method also linearly interpolates the antenna positions to align with the DMJDs corresponding to the correlations. It then returns the resulting correlation matrix, interpolated offset angles, FITS file meta data, raw offset angles, raw DMJD values, and DMJD values corresponding to the integrated correlations. The function declaration is as follows:

```matlab
function [ R, my_az, my_el, info, az_off, el_off, dmjd, dec_dmjd ] =... aggregate_banks( save_dir, ant_dir, tstamp, on_off, Nint )
```

One must specify a directory in which to save the resulting aggregated correlations and positions by setting `save_dir` to the desired directory. The directory that contains the antenna FITS files must also be specified by `ant_dir`. The time stamp (formatted as “`yyyy_mm_dd_hh:mm:ss`”) for the scan to aggregate must also be provided in `tstamp`. The `on_off` argument is deprecated, but should be set to 1 until downstream code is refactored. Lastly, one specifies the number of correlations per integration by setting `Nint`. If `Nint` is set to -1, all correlations in the scan are averaged.
aggregate_banks_pfb

This function does the same thing as aggregate_banks except that it expects fine-channel data from the fine-channel correlator. The function declaration is as follows:

```matlab
function [ R, my_az, my_el, info ] = aggregate_banks_pfb( save_dir, ant_dir, tstamp, Nint )
```

aggregate_banks_rb_hack

This function does the same thing as aggregate_banks except that it only aggregates data for a single frequency channel. This should only be used when wanting to get a quick look at, say, a sensitivity map for a single frequency bin. The function declaration is as follows:

```matlab
function [ R, my_az, my_el, info ] = aggregate_banks_rb_hack( save_dir, ant_dir, tstamp, on_off, Nint )
```

aggregate_single_bank

This function does the same thing as aggregate_banks except that it only aggregates data for a single BANK. This was done so as to minimize memory usage when processing long scans such as a Daisy scan. The function declaration is as follows:

```matlab
function [ R, my_az, my_el, xid, info ] = aggregate_single_bank( save_dir, ant_dir, tstamp, bank, Nint )
```

get_antenna_positions

This function reads an antenna FITS file and extracts the antenna positions. It also returns the DMJD values corresponding to the antenna positions and the RA/DEC angles. The function declaration is as follows:

```matlab
function [ dmjd, az_off, el_off, ra, dec ] = get_antenna_positions( fits_file, on_off, use_radec )
```

The argument on_off is deprecated and should always be 1. The argument use_radec returns the RA/DEC angles as if they were the azimuth and elevation offset angles.
**scan_table**

This is a script that contains information about each observation session. For example, a table relating the scan numbers with time stamps is generated, and element mappings (i.e., correlation entries to dipole number) are provided. Any dipoles that have failed or frequency channels that were lost are specified here as well. A new entry in the `scan_table` should be created for each observation session.

**source_table**

This is a script that contains information about calibration sources. Each entry is a struct that contains the source name and constants that are used to derive the source’s flux density across frequency.

**stitch_pfb**

This is a quick-and-dirty script that reads in some fine-channel correlations and generates a quick spectrum. Details of this script is beyond the scope of this work since it was developed by Mitchell Burnett.

**create_weight_file**

This function takes beam angles and weight values and generates a weight file. The exact file format is depicted in Figure B.8. The function declaration is as follows:

```matlab
function create_weight_file(az, el, wX, wY, cal_filename, X_idx, Y_idx, filename)
```

To use this function, one must also specify the mappings from dipoles to correlation matrix entry in `X_idx` and `Y_idx`.

**compute_delay**

This function estimates the delay between antenna signal paths by examining the phase structure across frequency. This was primarily used to evaluate the effectiveness of our timing alignment algorithm known as “word lock.” The function declaration is as follows:
function [ delta_n, residual ] = compute_delay(R, faxis, fs, ref_el)

One must specify the sample rate (fs), center frequencies for each bin (faxis), and to which element the delays will be relative (ref_el, correlation diagonal index). It then returns the sample offsets in delta_n and the residual error from the linear regression in residual.

Note: this function would likely be better placed in the word_lock directory.

B.6.2 sensitivity_maps

This sub-directory contains scripts and functions that generate sensitivity maps, trajectory plots, and calibrated steering vectors and beamformer weights.
plot_trjectories

This script extracts the antenna positions for a series of scans and plots the trajectory traversed by the telescope during those scans. One can also specify “off” or reference scans that are plotted with “x” markers.

sensitivity_map

This script computes the formed beam sensitivity for a series of scans in an observation session and plots them as a map. One must specify which session is being processed, the scan numbers for each of the on-source scans and reference (“off”) scans. One may also specify a note string that will differentiate the resulting calibration steering vectors and weight file names from other grids in the same observation session. The source must also be specified using one of the defined structs in source_table, and the number of correlation matrices to average per grid point. For example, if the correlations were saved with 0.1 second resolution, and the desired resolution of the grid is one second, Nint should be set to 10.

The script will then save the resulting calibrated steering vectors, maximum-SNR beamformer weights, and measured $T_{sys}/\eta$ values for each grid point. These files are then used when generating real-time beamformer weight files, pattern plots, and calibrated HI spectra.

sensitivity_daisy

This script does the same things as sensitivity_map except that it processes each BANK separately. All but the normalized system temperatures are then saved.

B.6.3 patterns

This sub-directory contains scripts and functions that generate beam or element patterns or assist with said generation.
get_beamformed_patterns

This method computes beam patterns using the calibrated steering vectors for a specified observation session, polarization, and set of weights. It then returns the pattern azimuth and elevation angles and pattern values. The function declaration is as follows:

\[ \text{function } [\text{AZ}, \text{EL}, \text{patterns}] = \text{get Beamformed Patterns}(\text{session}, \text{pol}, \text{note}, w) \]

Note: the note argument is used to distinguish between calibration sets from the same session, and is specified in sensitivity_map.

get_element_patterns

This is a script that plots element patterns for a specified session and polarization. This should eventually be converted into a function.

get_grid_steering_vectors

This function loads in the saved calibrated steering vectors generated when running sensitivity_map and returns the vectors closest to the specified azimuth and elevation offset angles. It also returns the angles corresponding to the return steering vectors and the measured normalized system temperature at those angles. The function declaration is as follows:

\[ \text{function } [\text{a, a_az, a_el, Tsys}] = \text{get Grid Steering Vectors}(\text{session}, \text{pol}, \text{note, beam_az, beam_el}) \]

get_grid_weights

This function loads in the saved maximum-SNR beamformer weights generated when running sensitivity_map and returns the weights for beams pointing in the user-specified directions. It also returns the beam pointing angle corresponding to the returned weight vectors. The function declaration is as follows:

\[ \text{function } [\text{w, w_az, w_el}] = \text{get Grid Weights}(\text{session, pol, beam_az, beam_el, note}) \]
plot_beam_patterns

This script will plot beam patterns for seven beams using weights and steering vectors generated from the same observation session’s calibration grid. It also generates a binary weight file that can be used in the real-time beamformer.

plot_hex

This function plots the beam patterns for a given observation session and set of beamformer weights in a hexagonal pattern. It then returns a handle to the generated figure. The function declaration is as follows:

function map_fig = plot_hex(session, AZ, EL, patterns)

plot_lcmv_patterns

This script generates linearly constrained minimum variance beamformer weights with several null constraints and plots the resulting patterns for the standard seven beam layout. It also generates a binary weight file that can be used in the real-time beamformer.

RTBF_data_analysis

Script that quickly looks at the total beamformed power in a time/frequency window. Currently this script is incorrectly placed in this sub-directory and would be better suited for misc.

B.6.4 tsys_plots

This sub-directory contains scripts and functions relating to generating system noise temperature spectra.

onoff_table

This script tabulates pairs of on/off source scans. One must specify which session the scans are from, the scan numbers for the two scans, the source struct from scan_table, and the LO frequency (Hz). This is then used in other scripts in this directory.
get_onoff_tsys

This function computes the normalized system temperature as a function of frequency for a given pair of on/off scans. One must specify the session struct from scan_table, the on/off scan numbers, the source struct from scan_table, and the LO frequency (Hz). The function declaration is as follows:

```matlab
function [Tsys_etaX, Tsys_etaY, freqs, wX, wY] = ...
    get_onoff_tsys(session, on_scan, off_scan, source, LO_freq)
```

The function returns the system temperatures for both polarization, the center frequencies for each frequency bin, and (for convenience) the maximum-SNR beamformer weights for a beam pointing in the direction of the on source.

plot_broad_tsys

This script compiles the system noise temperatures for a set of on/off pairs acquired with different LO frequencies and plots a broad spectrum. One must specify the session struct, the scan pairs, and the desired polarization.

run_all_onoff

This script gets the normalized system noise temperature for each pair of on/off scans specified in onoff_table. This is a useful script when trying to batch the aggregation process for every pair of on/off scans.

B.6.5 word_lock

This sub-directory contains scripts and functions relating to the “word lock” procedure, which attempts to time-align the element signal paths.

snoop_lock

This script reads in a coarse-channel correlation FITS file when observing noise from the noise source (i.e., the noise is coherent across elements) and estimates the sample offsets between
the various paths through a phase ramp analysis. The resulting sample offsets are then written to a file to be read later and applied in ROACH boards.

**word_lock**

This script does the same thing as *snoop_lock* except that it instead expects a series of FITS files that were acquired when observing a test tone at different frequencies.

**compare_lock**

This script evaluates the signal path time alignment after the sample offsets resulting from *snoop_lock* have been applied to the ROACH boards. One must specify which reference element was used in *snoop_lock* by changing `ref_el`.

**B.6.6 misc**

This sub-directory is meant to hold any quick-and-dirty scripts/functions that do not naturally fall into any of the other directories. At the time of this writing, there was only one script in this directory.

**peak**

This script reads in a single scan’s worth of data corresponding to a PEAK scan, in which the telescope drifts across a bright point source. It then plots the total power in the first element (correlation diagonal entry 0) in a single BANK against the reported antenna position to evaluate pointing alignment.
APPENDIX C. HASHPIPE HELLO WORLD

All of the following codes can be found at

https://github.com/rallenblack/hashpipe_samples.git.

C.1 hw_databuf.h

This code outlines the specifications for the inter-thread buffer.

```c
#ifndef _HW_DATABUF_H
#define _HW_DATABUF_H

#include "hashpipe_databuf.h"

/* Hello World for hashpipe!
* Author: Richard Black
* Date: Jan. 11, 2017
*
* Processing structure
* ======================================*
* hw_thread1 -> hw_buffer1 -> hw_thread2
* ======================================*
* thread1 will put the letters "HELLO WORLD!" into buffer1.
* thread2 will read buffer1 and print out contents
* This header file explains how the buffer is to be structured.
*/

// Macros to maintain cache alignment
#define CACHE_ALIGNMENT (128)
typedef uint8_t hashpipe_databuf_cache_alignment[
    CACHE_ALIGNMENT - (sizeof(hashpipe_databuf_t)%CACHE_ALIGNMENT)
];

// Number of blocks in the first buffer
```
#define N_BLOCKS1 5

// Create block header struct
typedef struct hw_buffer1_header
    int block_number; // We'll keep track of how many blocks we've processed
    hw_buffer1_header_t;

// Create dummy structure to make header size a multiple of CACHE_ALIGNMENT
typedef uint8_t hw_buffer1_cache_alignment[
    CACHE_ALIGNMENT - (sizeof(hw_buffer1_header_t)%CACHE_ALIGNMENT)
];

// Create the actual block struct
typedef struct hw_buffer1_block
    hw_buffer1_header_t header; // Put the header in the block
    hw_buffer1_cache_alignment padding; // Force data to be aligned
    char data[4]; // Three characters per block
    hw_buffer1_block_t;

// Create the full buffer
typedef struct hw_buffer1_databuf
    hashpipe_databuf_t header; // All hashpipe buffers must have this header
    hashpipe_databuf_cache_alignment padding; // Force buffer to be aligned
    hw_buffer1_block_t block[N_BLOCKS1];
    hw_buffer1_databuf_t;

 breathed synonym

* Create buffer control method prototypes
* (Definitions in hw.databuf.c)
*******************************************************************************/

/*
 * hw_buffer1_databuf_create
 * Creates an hw_buffer1_databuf_t hashpipe-compatible buffer
 * @arg int instance_id
 * The hashpipe instance id (set by -I on the command line)
 * @arg int databuf_id
 * The buffer's id -- typically set by hashpipe during startup
 * @return hashpipe_databuf_t *
 * A pointer to the newly created hashpipe-compatible buffer
 */
hashpipe_databuf_t * hw_buffer1_databuf_create(int instance_id, int databuf_id);
int hw_buffer1_databuf_wait_free(hw_buffer1_databuf_t * d, int block_id);

int hw_buffer1_databuf_wait_filled(hw_buffer1_databuf_t * d, int block_id);

int hw_buffer1_databuf_set_free(hw_buffer1_databuf_t * d, int block_id);

int hw_buffer1_databuf_set_filled(hw_buffer1_databuf_t * d, int block_id);
C.2 hw_databuf.c

This code defines the methods that were prototyped in hw_databuf.h.

```c
#include "hw_databuf.h"

hashpipe_databuf_t * hw_buffer1_databuf_create(int instance_id, int databuf_id)
    size_t header_size = sizeof(hashpipe_databuf_t) + sizeof(hashpipe_databuf_cache_alignment);
    size_t block_size = sizeof(hw_buffer1_block_t);
    int n_block = N_BLOCKS1;
    return hashpipe_databuf_create(
        instance_id, databuf_id, header_size, block_size, n_block);

int hw_buffer1_databuf_wait_free(hw_buffer1_databuf_t * d, int block_id)
    return hashpipe_databuf_wait_free((hashpipe_databuf_t *)d, block_id);

int hw_buffer1_databuf_wait_filled(hw_buffer1_databuf_t * d, int block_id)
    return hashpipe_databuf_wait_filled((hashpipe_databuf_t *)d, block_id);

int hw_buffer1_databuf_set_free(hw_buffer1_databuf_t * d, int block_id)
    return hashpipe_databuf_set_free((hashpipe_databuf_t *)d, block_id);

int hw_buffer1_databuf_set_filled(hw_buffer1_databuf_t * d, int block_id)
    return hashpipe_databuf_set_filled((hashpipe_databuf_t *)d, block_id);
```

C.3 hw_thread1.c

This code defines the first thread in the plugin.

```c
#include <pthread.h>
#include <string.h>
#include "hashpipe.h"
#include "hw_databuf.h"

/****************************
 * hw_thread1.c
 * Author: Richard Black
 * Date: January 12, 2017
```
This thread creates the string "HELLO WORLD!" and passes that string 3 characters at a time to hw_thread2 through the hw_buffer1_databuf_t buffer.

static void * run(hashpipe_thread_args_t * args)
{
    // Create a local pointer to the output buffer
    hw_buffer1_databuf_t * db_out = (hw_buffer1_databuf_t *)args->obuf;

    // Get access to status shared memory key/values
    hashpipe_status_t st = args->st;

    // Create string to print in the other thread
    const char test_word[13] = "HELLO WORLD!\0";

    // Create string counter
    int str_idx = 0;

    // Create block index tracker
    int block_idx = 0;

    // Create iteration counter
    int iter_count = 0;

    // Execute the main loop of the thread
    int rv;
    while (run_threads())
    {
        // Wait for the buffer block to be available for writing
        while ((rv=hw_buffer1_databuf_wait_free(db_out, block_idx)) != HASHPIPE_OK)
        {
            // If we time out, print "waiting" to status keyword "THREAD1"
            hashpipe_status_lock_safe(&st); // Gives us exclusive access to shared memory
            hputs(st.buf, "THREAD1", "waiting"); // Put string "waiting" with key "THREAD1"
            hashpipe_status_unlock_safe(&st); // Releases our exclusive access
        }

        // Once we get here, we have a ready-for-writing block in the buffer

        // Copy part of the string to the buffer
        strncpy(db_out->block[block_idx].data, test_word + (str_idx % 12), 3);

        // Update the string pointer
str_idx += 3;

// Write the loop counter
db_out->block[block_idx].header.block_number = iter_count;

// Only do this 20 times
if (iter_count < 20)
    // Update the loop counter
    iter_count++;

    // Mark block as filled so next thread can process the data
    hw_buffer1_databuf_set_filled(db_out, block_idx);

// Update the block index
// We use the % operator to make the block_idx circle back
    block_idx = (block_idx + 1) % N_BLOCKS1;

else
    hashpipe_status_lock_safe(&st);
    hputs(st.buf, "THREAD1", "Done!");
    hashpipe_status_unlock_safe(&st);

// Check to see if hashpipe is closing
pthread_testcancel();

return NULL;

// Thread description for hashpipe
static hashpipe_thread_desc_t hw_thread1 =
    name: "hw_thread1", // The name of the thread for the command line
    skey: "STAT1", // A shared memory keyword for thread status msgs
    init: NULL, // The name of the initialization function (NULL if none)
    run: run, // The name of the main loop function
    ibuf_desc: NULL, // The input buffer creation method name (NULL if none)
    obuf_desc: hw_buffer1_databuf_create // The output buffer creation method name (NULL if none)
;

// Hashpipe calls this method to enable this thread
static __attribute__((constructor)) void ctor()
    register_hashpipe_thread(&hw_thread1);
C.4  hw_thread2.c

This code defines the second thread in the plugin.

#include <pthread.h>
#include <string.h>
#include "hashpipe.h"
#include "hw_databuf.h"

/*****************************************************/
/* hw_thread2.c */
/* Author: Richard Black */
/* Date: January 12, 2017 */
/*
 * This thread waits for data to become available in
 * the hw_buffer1_databuf_t buffer. It then prints
 * out the contents of the buffer, which should be
 * three characters of the string "HELLO WORLD!"
 *
 * Expected output:
 * ----------------
 * 0: HEL
 * 1: LO
 * 2: WOR
 * 3: LD!
 * 4: HEL
 * 5: LO
 * 6: WOR
 * 7: LD!
 * ...
 * 19: LD!
 * ----------------
 *****************************************************/

// Run method for the thread
static void * run(hashpipe_thread_args_t * args)
{

    // Create a local pointer to the input buffer
    hw_buffer1_databuf_t * db_in = (hw_buffer1_databuf_t *)args->ibuf;

    // Get access to status shared memory key/values
    hashpipe_status_t st = args->st;
// Create placeholder for incoming data
cchar new_chars[4];

// Create iteration count placeholder
int iter_count = -1;

// Create block index tracker
int block_idx = 0;

// Execute the main loop of the thread
int rv;
while (run_threads())
    // Wait for the buffer block to be available for writing
    while ((rv=hw_buffer1_databuf_wait_filled(db_in, block_idx)) != HASHPIPE_OK)
        // If we time out, print "waiting" to status keyword "THREAD1"
        hashpipe_status_lock_safe(&st); // Gives us exclusive access to shared memory
        hputs(st.buf, "THREAD2", "waiting"); // Put string "waiting" with key "THREAD1"
        hashpipe_status_unlock_safe(&st); // Releases our exclusive access
        pthread_testcancel(); // Check if process is ending

    // Once we get here, we have a ready-for-processing block in the buffer
    // Copy buffer contents to local memory
    strncpy(new_chars, db_in->block[block_idx].data, 3);
    new_chars[3] = '\0';

    // Get the iteration count
    iter_count = db_in->block[block_idx].header.block_number;

    // Print the string to the console
    printf("%d: %s\n", iter_count, new_chars);

    // Mark block as free so it can get new data
    hw_buffer1_databuf_set_free(db_in, block_idx);

    // Update the block index
    block_idx = (block_idx + 1) % N_BLOCKS1;

    // Check to see if hashpipe is closing
    pthread_testcancel();
return NULL;

// Thread description for hashpipe
static hashpipe_thread_desc_t hw_thread2 =
  name: "hw_thread2",
  skey: "STAT2",
  init: NULL,
  run: run,
  ibuf_desc: hw_buffer1_databuf_create,
  obuf_desc: NULL
;

static __attribute__((constructor)) void ctor()
    register_hashpipe_thread(&hw_thread2);

C.5 Makefile.am

This is the required Makefile template needed for automake.

ACLOCAL_AMFLAGS = -I m4
AM_CPPFLAGS =

AM_CPPFLAGS += -I"@HASHPIPE_INCDIR@"

# AM_CFLAGS is used for all C compiles
AM_CFLAGS = -fPIC -O3 -Wall -Werror -fno-strict-aliasing -mavx

hw_databuf = hw_databuf.h |
    hw_databuf.c

hw_threads = hw_thread1.c |
    hw_thread2.c

# This is the flag_gpu plugin itself
lib_LTLIBRARIES = hw_hashpipe.la
hw_hashpipe_la SOURCES = $(hw_databuf) $(hw_threads)
hw_hashpipe_la_LIBADD = -lrt -L/usr/local/cuda/lib64
hw_hashpipe_la_LDFLAGS = -avoid-version -module -shared -export-dynamic --enable-shared
hw_hashpipe_la_LDFLAGS += -L"@HASHPIPE_LIBDIR@" -Wl,-rpath,"@HASHPIPE_LIBDIR@"
This is the required auto-configure script needed for automake.

# -*- Autoconf -*-
# Process this file with autoconf to produce a configure script.

AC_PREREQ([2.63])
AC_INIT([FULL-PACKAGE-NAME], [VERSION], [BUG-REPORT-ADDRESS])
AM_INIT_AUTOMAKE([foreign])
LT_INIT
AM_SILENT_RULES([yes])
#AC_CONFIG_SRCDIR([paper_databuf.h])
AC_CONFIG_HEADERS([config.h])
AC_CONFIG_MACRO_DIR([m4])

# Set CFLAGS to nothing if it is not set by the user. This prevents AC_PROG_CC
# from setting the (supposedly reserved-for-the-user!) variable CFLAGS in
# Makefile, which prevents AM_CFLAGS in Makefile.am from setting an
# optimization level. For more details, see
AS_VAR_SET_IF(CFLAGS,[],[CFLAGS=])

# Checks for programs.
AC_PROG_CC

# Check for HASHPIPE and xGPU and total_power
AX_CHECK_HASHPIPE
AX_CHECK_XGPUINFO
AX_CHECK_FLAGBEAMFORM
AX_CHECK_FLAGPOW

# Checks for libraries.
AC_CHECK_LIB([pthread], [pthread_create])
AC_CHECK_LIB([rt], [clock_gettime])
AC_CHECK_LIB([z], [crc32])

# Checks for header files.
AC_CHECK_HEADERS([netdb.h stdint.h stdlib.h string.h sys/socket.h sys/time.h unistd.h zlib.h])

# Checks for typedefs, structures, and compiler characteristics.
AC_C_INLINE
AC_TYPE_INT32_T
AC_TYPE_INT64_T
AC_TYPE_OFF_T
AC_TYPE_SIZE_T
C.7 hashpipe.m4

This script looks for the installed HASHPIPE libraries so the plugin can be installed.

serial 1 hashpipe.m4
AC_DEFUN([AX_CHECK_HASHPIPE],
[AC_PREREQ([2.65])]
AC_ARG_WITH([hashpipe],
  AC_HELP_STRING([--with-hashpipe=DIR],
  [Location of HASHPIPE files (/usr/local)],
  [HASHPIPEDIR="$withval"],
  [HASHPIPEDIR=/usr/local])
orig_LDFLAGS="$LDFLAGS"
LDFLAGS="$orig_LDFLAGS -L$HASHPIPEDIR/lib"
AC_CHECK_LIB([hashpipe], [hashpipe_datbuf_create],
  # Found
  AC_SUBST(HASHPIPE_LIBDIR,$HASHPIPEDIR/lib),
  # Not found there, check HASHPIPEDIR
  AS_UNSET(ac_cv_lib_hashpipe_hashpipe_datbuf_create)
  LDFLAGS="$orig_LDFLAGS -L$HASHPIPEDIR"
  AC_CHECK_LIB([hashpipe], [hashpipe_datbuf_create],
    # Found
    AC_SUBST(HASHPIPE_LIBDIR,$HASHPIPEDIR),
    # Not found there, error
    AC_MSG_ERROR([HASHPIPE library not found])))
LDFLAGS="$orig_LDFLAGS"

AC_CHECK_FILE([$HASHPIPEDIR/include/hashpipe.h],
  # Found
  AC_SUBST(HASHPIPE_INCDIR,$HASHPIPEDIR/include),
  # Not found there, check HASHPIPEDIR
  AC_CHECK_FILE([$HASHPIPEDIR/hashpipe.h],
# Found

AC_SUBST(HASHPIPE_INCDIR,$HASHPIPEDIR),

# Not found there, error

AC_MSG_ERROR([hashpipe.h header file not found]))}