UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

METHODS FOR CONTROL, CALIBRATION, AND PERFORMANCE OPTIMIZATION OF PHASED ARRAY SYSTEMS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

MATTHEW MARTIN HERNDON
Norman, Oklahoma
2022

METHODS FOR CONTROL, CALIBRATION, AND PERFORMANCE OPTIMIZATION OF PHASED ARRAY SYSTEMS

A THESIS APPROVED FOR THE

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY THE COMMITTEE CONSISTING OF

Dr. Mark Yeary, Chair

Dr. Robert Palmer

Dr. Caleb Fulton

© Copyright by MATTHEW MARTIN HERNDON 2022
All Rights Reserved.

Acknowledgements

To my parents, brother, and sister, for supporting me throughout my academic career. To my advisor, Mark Yeary, for his continuous guidance and patient support throughout my time in this program. And to the staff engineers, faculty, and leadership at the Advanced Radar Research Center for helping me grow as an engineer, person, and professional.

Contents

List of Figures

Abstract

Phased array radar systems have proven advantageous in a variety of research applications, offering faster volume scans and unparalleled time-resolution as compared to traditional parabolic dish antenna systems that rely solely on mechanical systems for controlling the direction of radiation. As such, research has accelerated the development of practical phased array systems to realize their full vision. In particular, next generation phased array systems aim to provide additional advantages in the form of reconfigurable beam patterns, adaptive digital beamforming, multiple-input multiple-output (MIMO) radar modes, and other software-defined technologies. However, to fully realize a paradigm shift in phased array technology, especially as the ratio of array to sub-array size becomes greater, this requires a corresponding increase in novel digital backend architectures to fully achieve this vision. Therefore, new methods for control, calibration, and performance optimization are required to enable next-generation phased array systems to reach their potential. In this thesis, a variety of practical engineering challenges related to phased array system design are discussed, with system-level implications and relevant theory included where necessary. For instance, for the first time, as explained in this thesis, a GPS disciplined, time-interleaved measurement technique that leveraged real-time control of a beamformer was developed to enable accurate post-processing correction of the phase drift that results from clocking differences between noncoherent physically separated bistatic nodes. In addition, laboratory efficacy of digital predistortion using the memory-polynomial model has been confirmed for the purpose of maximizing an element's usable power while minimizing spectral spreading and achieving desirable output linearity during operation, and a novel method for training predistortion models comprised of a combined software-defined

and physical mechanism for measuring transmitter front-end distortion for elements within a digital-at-every element array has been proposed and verified in the lab.

Chapter 1

Introduction

With the development of high-speed mobile networking and cloud computing, critical RF and digital logic components—namely digital transceivers and to a somewhat lesser degree, FPGA systems-on-a-chip (SoC)—have rapidly become infrastructure technologies essential to modern life. The race towards these new paradigms has motivated research into these two critical backend technologies, helping to place downward pressure on chip costs and improving ease-of-development as they achieve widespread industry adoption. Digital transceiver SoCs collectively drive network access for billions of devices globally, with FPGAs often providing a stable, high-performance platform for implementing control logic and algorithms in these wireless systems. Likewise, compiler and co-processing technologies such as high-level-synthesis (HLS) and OpenCL simplify certain aspects of FPGA development, making it feasible to deploy FPGAs in cloud compute farms for accelerating workflow computation. These innovations have empowered parallel industries and research areas to adopt these chips into their systems—all-the-while inheriting best practices via direct knowledge transfer—improving their capabilities and motivating the development of new control, calibration, and optimization methods which advance their unique state-of-the-art.

In particular, phased array radar (PAR) systems—formerly confined to the defense space owing to their high costs and technical complexity—are seeing their critical components rapidly fall in cost, making it financially and logistically feasible from an engineering standpoint to develop radar research platforms leveraging these technologies. PAR systems offer many benefits over more traditional, exclusively mechanically steered systems, including electronic beam control enabling faster volume scans, adaptable beam patterns, and other benefits [54, 62]. Systems may also be designed following a modular architecture with improved failure tolerance, wherein high-power front-end components exposed to the most stress (those most likely to fail) are distributed behind either each array element or behind sets of elements (i.e. sub-arrays) in modules, which pieced together form the larger array. When failures inevitably occur, they are quarantined within their defective modules, allowing the larger system to continue functioning with reduced but still viable performance until repairs are completed. Next-generation PARs leveraging digital-at-every-element technology will also offer additional advantages in the form of digital beamforming, multiple-input multiple-output (MIMO) radar, broad reconfigurability by virtue of their FPGA-based digital back-ends, and many other benefits—each made possible or practical by the unmatched flexibility of such systems.

However, to fully realize this paradigm shift in technology requires a corresponding increase in technical complexity when implementing such systems, requiring complex and novel architectures to achieve their vision. As such, new methods for control, calibration, and performance optimization are required to enable next-generation phased array systems to reach their potential.

1.1 Organization

In this thesis, the author's contributions to several radar projects undertaken by the Advanced Radar Research Center (ARRC) are described in context along with relevant theory and background. These contributions range in complexity and scope, but each ultimately emphasizes specific techniques used to control, acquire meaningful measurements from, or calibrate and/or optimize the performance of a radar system. In each case, these practical concerns are given theoretical justification and contextualized by examining their system-level implications, where relevant. As such, much of the efforts described therein aim to serve as case studies in applied engineering, with much of the content focused on the practical engineering challenges faced during the design process.

In Chapter 1, a software suite for controlling radar systems is described along with its design philosophy and motivating principles. This software—originally developed in support of a single project—was adapted and grown into a generalized platform for controlling experimental hardware, and was leveraged for much of the work described in this thesis. Chapter 2 is concerned with phase synchronization in a bistatic radar system subject to varying phase drift, and in particular describes how an interleaved measurement technique leveraging real-time control of an analog beamformer was developed to enable accurate drift correction. Additionally, this chapter includes discussion of how alignment calibration was performed by deriving array calibration weights from drift-corrected element patterns. Lastly, Chapter 3 discusses digital predistortion in the abstract along with a system architecture for applying the algorithm to data streams in real-time. This method seeks to improve system performance by linearizing a high-power amplifier's gain response both in compression and across a bandwidth, with the aim of maximizing the amplifier's power output without introducing distortion

typically associated with nonlinear amplifiers operating in saturation.

Chapter 2

Distributed Multi-static Radar Control via RadarControlGUI and RadarControllerAPI

The development of radar systems requires sequenced control of many different—often independent—subsystems, including components responsible for RF synthesis, beamforming (in the case of phased arrays), synchronous triggering mechanisms, and mechanical pedestals, among others. Without strict controls in place, the interplay between devices in such systems can introduce software bugs which limit the repeatability of experiments and correspondingly impact the research process. Effective research and development in this space therefore requires resolving this complexity in some capacity.

For radar systems in particular, a variety of common software problems exist which either must be solved in order to accomplish a particular set of research goals or alternately serve to accelerate the pace of research by simplifying aspects of the control process. During the initial design phase, several core features were targeted as essential for enabling effective control, including methods for

- providing asynchronous, networked access to resources,
- capturing and saving data to nonvolatile memory,
- providing connected subsystems with a means of publishing status and log-

ging information,

- synchronizing controls and state management across systems of varying topologies.

Likewise, intuitive user-facing software was an essential requirement for the software system, as this would act as the user's primary means of controlling the system. For an effective user experience, critical features of the graphical user interface include

- control and status widgets for interacting with the system,
- methods for visualizing data in real time,
- a means of sequencing system-level actions for running automated experiments in a repeatable manner.

With these criteria in mind, a software suite was developed with the goals of accelerating radar system development and simplifying scientific research for users of the system. In this chapter, this software suite will be discussed in abstract, with implications for later chapters which discuss systems which leveraged the software to accomplish their particular research goals.

2.1 Software Architecture

To provide a generalized control and data visualization platform for radar projects, software was developed to enable complex, repeatable control of systems via a flexible, user-friendly interface. This software was designed with the primary goal of enabling control of experimental weather radar systems in their capacity as both research platforms for capturing weather measurements and as a base for aiding in the calibration, development, and debugging of the systems—in short,

to provide sufficiently fine-grained control without being too tedious for operators. Two complementary and generalized software systems—*RadarControlGUI* and its corresponding backend *RadarControllerAPI*—were developed in tandem to enable control of multi-nodal network-distributed systems with the aim of accelerating the development of experimental radar systems by defining a standard architecture for creating graphical user interfaces (GUIs) with complex control capabilities. For additional reference, Fig. 2.1 diagrams the architecture of the software suite by visualizing the relationship between system components.

2.1.1 RadarControllerAPI

RadarControllerAPI defines basic support and control logic which standardizes and implements the core functionality of the software. Software systems designed following this framework are called *platforms* and consist of one or several *node(s)* connected via a single parent *system controller*.

Node

- *Nodes* are asynchronous endpoints defining controls for self-contained devices capable of exercising exclusive control over some set of subsystems.
- *Control fields* (i.e. variables with metadata) provide the primary means of managing system controls and status.
- State tracking for so-called *actions* with sequencing. *Actions* serve as an abstraction representing potentially long-duration events which exercise exclusive control over some part of the system, e.g. pedestal movements or transmit and receive operations. These form the basis of system control.

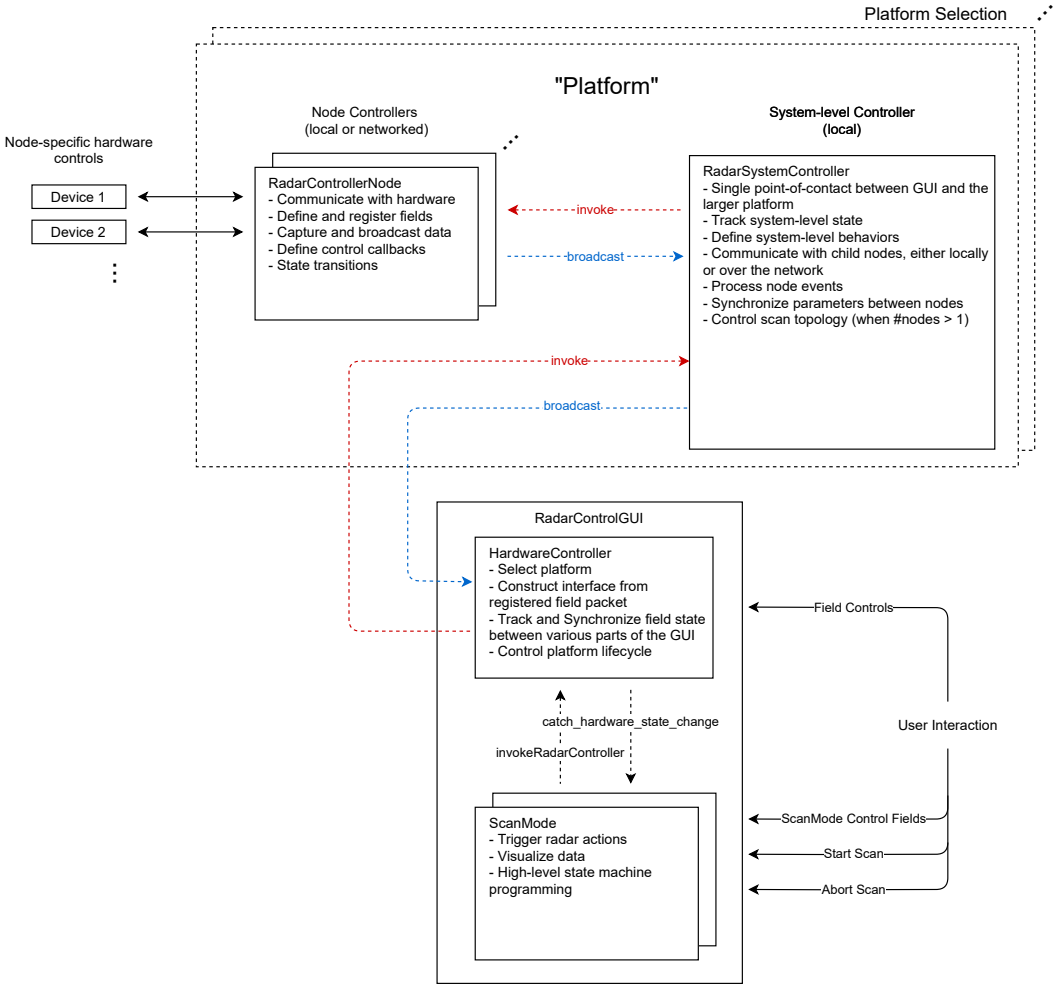


Figure 2.1: Diagram of the relationship between software system components.

- Transactional *invoke / broadcast* access methods provide a non-blocking means of triggering and obtaining responses from nodes. Additionally, the *broadcast* interface provides a formal interface for nodes to broadcast status updates, log messages, data, and control field updates to parent nodes to be interpreted as desired. All data passed through these interfaces must by serializable by necessity, a constraint which ensures that control may be maintained either within a single program instance or across a network connection. When networked, this scheme is easily expressed via a singly-

maintained bidirectional socket of the message-passing interface Nanomsg, allowing messages to be simultaneously sent and received.

System Controller

- A special *node* which fills a mediary role, managing and synchronizing control between one or several child nodes while also providing an interface for defining system-level behaviors (e.g. managing multistatic topologies).

2.1.2 RadarControlGUI

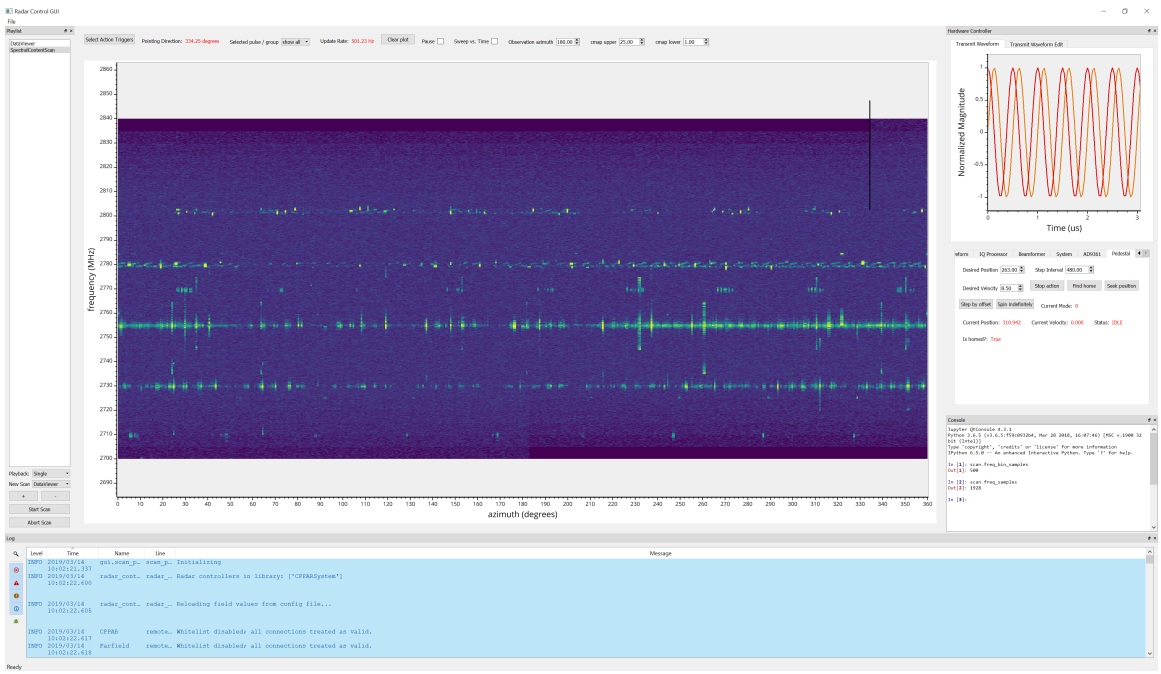


Figure 2.2: RadarControllerGUI example, showing a ScanMode that was developed to visualize the direction and frequency of interference observed from a radar system.

Although in theory platforms developed within RadarControllerAPI are operable in isolation, they were primarily designed to interface with a graphical user interface of some kind. For this reason, an externally maintained software sys-

tem, RadarControllerGUI, was developed to provide a user interface for systems developed using the API.

- Automatically generates a widget-based interface for controlling the selected *platform*. The GUI is capable of multiplexing between one or several platforms, connecting to each using its *system controller* as a single-point-of-contact.

- *ScanModes* provide a standardized way to: organize and enable the selection between different programs; receive, interpret and plot data; and trigger and respond to actions. This abstraction includes a callback-based scheme for interacting with asynchronous platform actions, allowing long-duration events to be triggered and awaited (and by extension, chained together) in order to construct complex experiments. In addition to several predefined ScanModes compliant with platforms by default (e.g. DataViewer and PowerByAzimuth, named to described their primary function), this abstraction was designed to sequester platform-specific code for visualizing, controlling, and interpreting data into small, manageable code snippets.

Of particular note, ScanModes provide a simple way of programming the system at a high level, and therefore offer a lot of utility during the research and development process. As an example, a ScanMode was developed to visualize interference observed by the system over some bandwidth. Fig. 2.2 shows RadarControllerGUI while running the interference detection ScanMode. The ScanMode defines an asynchronous procedure which changes the system's center frequency each time the pedestal's azimuthal position wraps around 0.00°, which indicated that it had completed one full rotation, and continues stepping through this bandwidth indefinitely. Incrementing this center frequency

for several iterations allowed an arbitrary bandwidth to be visualized automatically without being limited by the transceivers' static 30.00 MHz of instantaneous bandwidth. Likewise, performing all control actions using RadarControllerAPI's asynchronous invoke / broadcast scheme (described in Section 2.1.1) ensures that the GUI can continuously update throughout the scan, updating the interference map in real time.

2.2 Closing Remarks

This software suite was critical for accomplishing much of the research described in this thesis, and therefore provides important context related to system-level control of experimental radar systems. In the chapter that follows, the discussion will migrate to practical system control and calibration, and will begin by detailing calibration, control and measurement procedures for a cylindrical phased array system using a bistatic pattern measurement scheme.

Chapter 3

Pattern Measurement of a Cylindrical Array via Non-coherent, Bistatic Synchronization and Corresponding Channel Equalization

Figure 3.1: CPPAR and its radome installation on the roof of the Radar Innovations Laboratory.

For weather radar applications, PAR research has primarily focused on planar array architectures, seen in projects such as the single-polarization Atmospheric Imaging Radar (AIR) [31] at the University of Oklahoma (OU), the Skylar phased

array radar program at the University of Massachusetts developed in partnership with Raytheon [22], the dual-polarized Horus program at OU [46, 63], the Advanced Technology Demonstrator (ATD) [23, 28] operated by the National Severe Storms Laboratory in Oklahoma, and the Polarimetric Atmospheric Imaging Radar (PAIR) at OU [50]. Though these systems were designed to offer many of the advantages of PAR systems, they are limited in several key ways as a result of their planar geometry. In planar arrays, off-broadside beam steering reduces the effective aperture size of the array along steered dimension(s), widening formed beams and reducing the array's sensitivity in those positions. In dual-polarized weather radars, this effect also leads to marked reductions in the polarimetric purity of measurements taken from off-broadside beam positions, as a result of the inconsistency between the projected aperture sizes of the two polarizations in those positions. Additionally, for single-faced planar PARs to provide 360° of coverage in azimuth, they must mechanically rotate (as with parabolic dish radars), introducing azimuthal beam smearing and limiting their speed and flexibility when scanning the sky. Alternatively, planar PARs may leverage multiple array faces to provide full azimuthal coverage, but this inevitably creates *cones of silence* (or, zones of lower sensitivity) at boundaries between faces, which effectively results in systems with a maximum range that varies with respect to azimuth. These challenges faced by planar phased arrays have motivated research into alternative geometries without such drawbacks.

In contrast to planar arrays which must steer off broadside when electronically scanning, cylindrical arrays uniquely allow all azimuthal positions to be observed using broadside beams [13, 18, 30, 33, 38, 59, 64]. In cylindrical arrays, traditional beam steering is replaced by beam *commutation* as the primary mechanism for electronically steering the array in azimuth space. Commutation

refers to the method by which the direction of the array's observation beam is changed by electronically selecting which array elements are excited. In this scheme, each commutative position derives its directionality from its position within the array, rather than from a phase gradient altering the direction of the wavefront formed by the array (as with planar arrays). Each commutative position forms a broadside beam pointing from the center of the array through the center of the sector, which given the array's cylindrical geometry corresponds to a unique azimuthal pointing angle. The set of all possible commutative positions (determined by the number of array elements and the range of azimuthal positions spanned by the elements) forms broadside beams subdividing azimuth space which are then electronically selected between to *steer* in azimuth. The resulting all-broadside performance makes the cylindrical array geometry particularly attractive for weather radar, as it theoretically ensures consistent polarimetric purity across all electronic beam positions. In this way, cylindrical arrays offer one possible avenue for next-generation phased-array weather radars by offering an architecture which bypasses some weaknesses inherent to planar arrays [65]. Cylindrical arrays have also found use within 5G applications [20, 37].

Cylindrical arrays offer their own difficulties; namely, calibration is particularly challenging [4, 21, 35], as is beam synthesis [29, 56]. A lack of clear answers regarding the efficacy of dual-polarized phased array weather radars in general, cylindrical array design and beam generation motivated the Cylindrical Polarimetric Phased Array Radar (CPPAR) project, which acts as testbed for researching solutions to these challenges [19]. Several recent papers such as [39, 40] detail research conducted using the system. As seen in Fig. 3.1, the CPPAR system is mounted on top of the Radar Innovations Laboratory at the University of Oklahoma's South Research Campus in Norman, Oklahoma.

The CPPAR demonstrator's antenna array consists of 96 columns of 19 dual-polarized, frequency-scanned, aperture coupled, and stacked patch antennas that are designed to operate from approximately 2.70 to 3.00 GHz [32]. The radar's aperture is 2.00 m tall and 1.00 m in radius. The array is operated at frequency of 2.76 GHz so that the frequency-steering antenna columns of the array radiate at an offset of 3.30° in elevation. To enable flexible and rapidly-switchable azimuthal beamforming, the system's array is fed by four isolated analog beamforming networks controlled via FPGA logic developed at the ARRC. A single two-channel digital transceiver (Analog Devices, part AD9361) provides separate feeds for the transmitter and receiver of each polarization, with each of these four connections connecting to one of the four beamformers. At the CPPAR's Tx frontends, each polarization on each column is driven by an 80.00 W class C GaN amplifier, connected to the antenna through high-power T/R switches and low-loss RF coaxial cables.

The CPPAR system was developed specifically to enable in situ measurement of the array's far-field radiation patterns, both to facilitate calibration and to provide a platform for testing beam weight optimization algorithms. In this capacity, a separate two-channel radar system was developed to act as a bistatic calibration node. To enable time-aligned power and quasi-coherent phase measurements between the two nodes, the two nodes' S-band digital transceivers and deterministic FPGA controllers were each supplied with GPS-disciplined reference clocks and synchronous pulse-per-second triggers. Two horn antennas were mounted on the nearby National Weather Center (in the array's farfield) and oriented along its horizontal and vertical polarization planes. Circulators prior to these horns' feeding points allow the calibration node to operate as either a transmitter or receiver. To allow arbitrary positioning in azimuth when mea-

Figure 3.2: CPPAR and the far-field measurement probe (horn shown in the foreground of the picture) [26]. Reprinted from Herndon and Yeary (2021) © 2021 IEEE

suring patterns, the CPPAR itself was mounted on a mechanical pedestal which may be freely rotated around its central axis. Given the position of the horns with respect to the array, rotating the CPPAR in azimuth sweeps the CPPAR's beam approximately at boresight through both horns' beams, making it possible to measure characteristic far-field patterns. A diagram of this arrangement is shown in Fig. 3.2.

GPS has been used in the past to calibrate the phase of a radar [48, 49, 51]. Yet, this work differs since we have exquisite control over every column (enabling

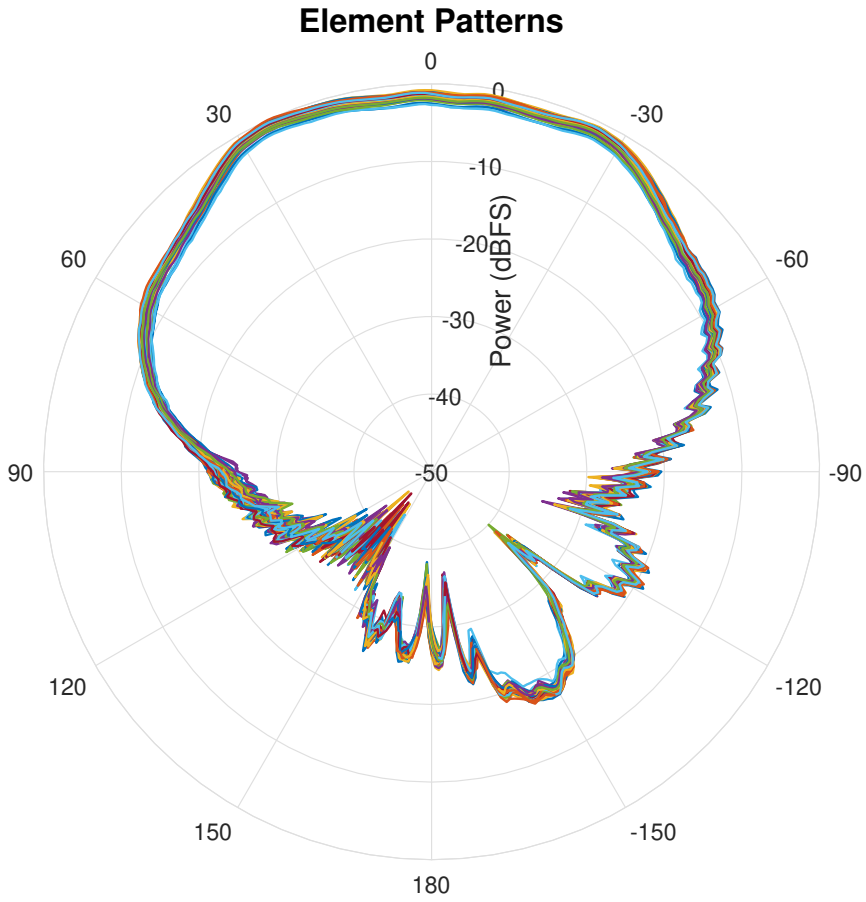


Figure 3.3: Spatially aligned power patterns for each of the CPPAR's 48 active elements, as measured from the far-field calibration horn.

rapid electronic steering in azimuth) and since the array can be mechanically rotated. Our method successfully combines both electronic and mechanical azimuth control into a monolithic scheme for capturing element patterns from the array in situ—a feat accomplished by rotating the pedestal while sampling patterns via interleaved electronic excitations of the array elements [26]. Precise accounting of this measurement's sampling geometry and two reinterpretations of the data this operation produced allowed for the effects of phase drift to be accurately measured and corrected for in post-processing. Ultimately, our technique reliably produced precise power (see Fig. 3.3) and phase (further explanation required) pattern measurements for all array elements from a single process lasting less

than one minute, leading to detailed array error measurements and ultimately enabling array calibration.

3.1 Overview of CPPAR System and Methods

3.1.1 System Architecture

The CPPAR system was developed specifically to allow for the on-site capture of the array's far-field radiation patterns to facilitate calibration as well as to provide a source of training data for iterative beam weight optimization algorithms. To allow arbitrary positioning in azimuth, the CPPAR was mounted on a mechanical pedestal which may be freely rotated around its central axis. To provide a means for measuring array patterns, two horn antennas oriented along the CPPAR's horizontal and vertical polarization planes were mounted on the nearby National Weather Center, directed towards the array, and supplied with a functional transmitter / receiver system analogous to the CPPAR's internal hardware to provide either transmit-mode excitations or receive-mode baseband sampling. A diagram of this arrangement is shown in Fig. 3.2. To enable time-aligned power and quasi-coherent phase measurements, the bistatic system's two S-band digital transceivers and deterministic FPGA controllers were supplied with GPS-disciplined reference clocks and synchronous pulse-per-second triggers. Given the position of the horns with respect to the array, rotating the CPPAR in azimuth sweeps the CPPAR's beam approximately at boresight through both horns' beams, making it possible to measure characteristic far-field patterns.

Likewise, to enable repeatable programmatic control of the bistatic system, its various hardware elements and their corresponding software controls were

connected together using RadarControllerAPI, the software suite described in Chapter 2. Both bistatic nodes within the system include a processing computer which centralizes aspects of their control behind a RadarControllerAPI 'node' running within a software daemon. These nodes are linked together over the network as children of the RadarControllerAPI 'platform' running as the root node within the RadarControllerGUI graphical user interface (GUI), which synchronizes control of the system and processes and visualizes its data returns. This software backend proved vital for ensuring stable and repeatable operation of the system.

3.1.2 Calibration Criteria

To define an effective calibration criteria, potential sources of error within the array must be identified and understood in order to form a sufficiently accurate error model. In CPPAR, the array is driven by a series of digitally controllable analog beamformer networks feeding to and from a single two-channel transceiver chip. Given the coarse accuracy of CPPAR's phase shifters and attenuators, sufficiently small effects (e.g. biases residing within the feeding network itself) are considered negligible when compared to the large variance resulting from differences in front-end electronics. This implies that the largest source of excitation variance occurs at the element level, and may thus be treated as independent. Following this assumption, a model was defined which relates each element's power and phase patterns via linear offsets to their 'spatially coherent', point-to-point mean.

The model is defined by $\mathrm{P}_n(\theta)$ and $\phi_n(\theta)$, which represent the power and phase patterns, respectively, measured at some azimuthal position θ and offset

from the element's boresight for a given element n. The corresponding equations are

$$\mathrm{P}_n(\theta) = |\mathrm{A}(\theta)| + P_n^{\mathtt{bias}} + \epsilon_p \tag{3.1}$$

and

$$\phi_n(\theta) = \angle \mathrm{A}(\theta) + \phi_n^{\mathtt{bias}} + \epsilon_\phi \quad . \tag{3.2}$$

Here, $\mathrm{A}(\theta)$ represents the baseline phased element pattern, $\hat{P}_n$ and $\hat{\phi}_n$ are the element's *true* power and phase bias terms which, and each ϵ is a random variable representing measurement noise. Estimating the offset terms $\hat{P}_n$ and $\hat{\phi}_n$ makes way for the derivation of a set of calibration terms, $\hat{P}_{cal_n}$ and $\hat{\phi}_{cal_n}$, which equalize power and phase across the array by biasing each element 'n' such that variance across the array is minimized.

For CPPAR, *ideal* calibration is defined as the state wherein the phases and powers of all elements are equalized with respect to their boresights. When this condition is met, inter-element power and phase variance is minimized with respect to a cylindrical alignment surface centered on CPPAR. In more practical terms and drawing directly from the error model defined by equations (3.1) and (3.2), the array is considered calibrated when (separately for both power and phase) the element pattern variance is minimized with respect to the average pattern. When these criteria are met, the array generates a radially symmetric excitation, which given the array's geometry is analogous to a planar array's uniform taper response.

One advantage this calibration criteria uniquely offers to cylindrical arrays is that it presents a straightforward path for the implementation of beam commutation. Derived only from the array geometry, a single sector of beamforming weights may be formed to generate a beam under ideal circumstances—as

the array is radially symmetric, these weights may then be commutated to any other position in the array to form approximately identical beams. Calibration weights are tied to specific elements, so theoretically they may be applied transparently at each commutative beam position. Assuming sufficiently accurate calibration weights and high-quality beam weights, this procedure yields many identical beams pointing in many directions without requiring individual optimization at for each set.

Understanding this, two disparate calibration goals were defined to inform the system's design:

- Element-level calibration, wherein element biases are corrected for to allow some ideal set of weights to be shared by all commutative beam positions; and

- Sector calibration, in which each beam position is optimized separately. This method provides the greatest control over the quality of beamforming, but requires substantially more effort.

In support of these objectives, the system's beamformer control logic was designed to allow phase and attenuation to be uniquely specified for each of the CPPAR's elements on a pulse-to-pulse basis. This allows complex scans ranging from commutative weather scans with varying pulse dwells to the time-interleaved capture of element patterns, a measurement necessary to characterize the system for use during calibration and pattern optimization.

3.1.3 Pattern Measurements

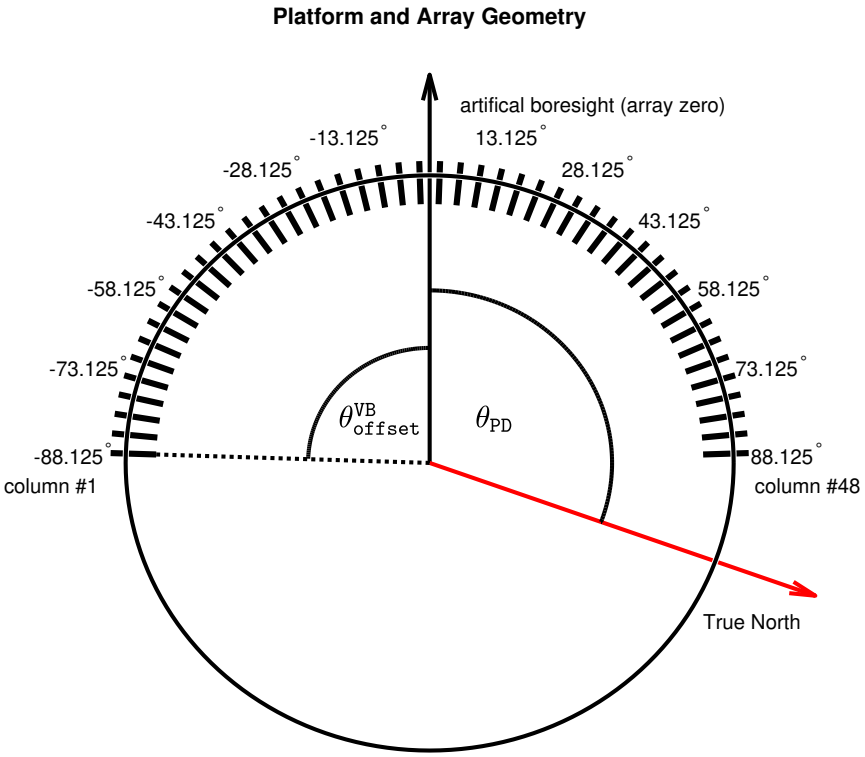


Figure 3.4: Element layout and azimuthal offset definitions for the CPPAR platform [26]. Reprinted from Herndon and Yeary (2021) © 2021 IEEE

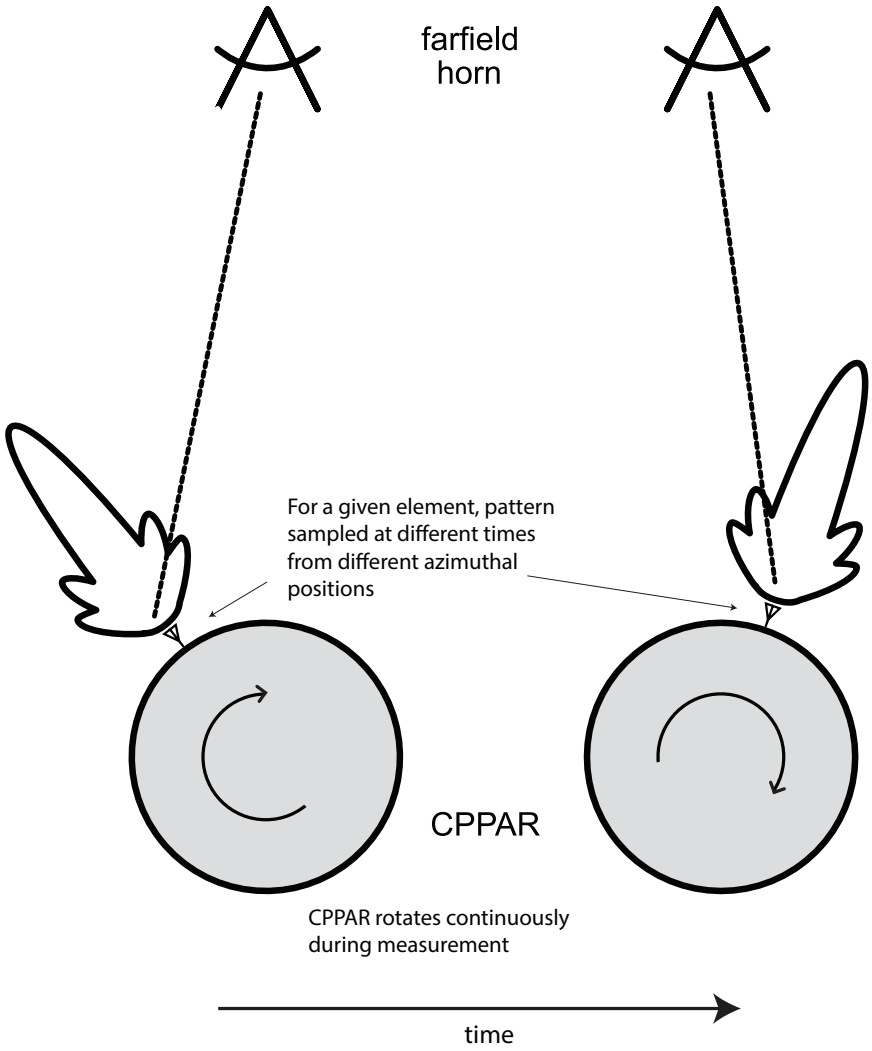


Figure 3.5: Transmit pattern measurement of a single element using CPPAR and the stationary far-field node [26]. Reprinted from Herndon and Yeary (2021) © 2021 IEEE

Before patterns could be captured and understood, the array's local coordinate system and its relationship to the surrounding environment was necessarily considered. Relative to the platform itself, the array's 96 antenna elements are arranged radially (see Fig. 3.4) with a spacing of $\theta^{\mathrm{EL}}_{\mathtt{spacing}} = \frac{360.00^\circ}{96} = 3.75^\circ$ in azimuth, however only half ($\mathrm{N}_{\mathtt{elements}} = 48$) of these elements are populated with electronics. Although in principle, cylindrical arrays have no fixed boresight as this reference point changes according to the actively selected commutative sec-

tor, it is useful to define an absolute reference point for the array *analogous* to boresight for simplified position tracking. This *virtual boresight* reference position was chosen arbitrarily as the center of the populated sector, located between the array's 23rd and 24th column, and is offset from the first populated column's center by $\theta^{\texttt{VB}}_{\texttt{offset}} = 88.13^\circ$ in azimuth, as expressed by

$$\theta^{\texttt{VB}}_{\texttt{offset}} = \frac{180^\circ}{2} - \frac{3.75^\circ}{2} = 88.13^\circ \quad . \tag{3.3}$$

Next, azimuthal offset terms relating each element's position to the virtual boresight position were defined as

$$\begin{aligned} \theta^{\texttt{EL}}_{\texttt{offset}} &= (n-1) \cdot \theta^{\texttt{EL}}_{\texttt{spacing}} - \theta^{\texttt{VB}}_{\texttt{offset}} \\ &\qquad \text{where } n \in \{1,\ 2,\ \cdots,\ 48\} \\ &= \{-88.13^\circ, -84.38^\circ,\ \cdots, 88.13^\circ\} \quad . \end{aligned} \tag{3.4}$$

In general, commutative beams may be tagged with an azimuthal offset which establishes its position with respect to the artificial boresight, allowing their absolute position to be computed with ease.

The mechanical pedestal holding the CPPAR allows the entire array to rotate continuously around its central axis with absolute position control. During operation, the pedestal's current azimuthal position is tracked in real time and made available in all data captured from the system. The system's *pointing direction* ($\boldsymbol{\theta}_{\texttt{PD}}$) is defined as the absolute azimuthal position of the aforementioned virtual boresight at any given point in time (see Fig. 3.4). This field also acts as the primary means of tracking the array's position, with each pulse group produced by the radar containing an estimate of the pointing direction at the time of capture.

With these features in mind, the most straightforward method for capturing

patterns with the CPPAR was to rotate the pedestal at a constant rate while periodically exciting the CPPAR's two-node bistatic system in one of two topologies:

1. `CPPAR`[`Tx`] $\rightarrow$ `Farfield`[`Rx`] for transmit patterns, or
2. `CPPAR`[`Rx`] $\leftarrow$ `Farfield`[`Tx`] for receive patterns.

To measure the CPPAR's receive patterns, the far-field calibration horn was configured to transmit a test signal which could then be observed by the CPPAR node operating as the receiver. Alternately for measuring the CPPAR's transmit patterns, the CPPAR transmits some test signal to be observed by the farfield node. In both topologies, these samples in combination with the array's position estimates provide a view of the array's far-field patterns under either mode when the array is excited with a given set of weights. Given the system's pulse-to-pulse beam selecting ability, many sets of weights may be interleaved together into a single scan, allowing multiple patterns to be measured nearly simultaneously. Leveraging this feature, a scan was devised to measure the array's $\mathrm{N}_{\texttt{elements}} = 48$ horizontal and vertical element patterns.

For each active element in the array, a *beam* (i.e. single set of full-array weights) was defined which excites the focused element and deactivates all others. Deactivated elements were configured with:

- a maximum attenuation of $-32.00\,\mathrm{dB}$ applied,
- fully disabled transmit path, with excess energy routed into a $50.00\,\Omega$ matched load rather than into the element's high-power amplifier, and lastly
- a disabled high-power amplifier trigger.

These conditions minimized the amount of energy radiated from the deactivated elements and ensured element patterns could be accurately measured in isolation from one another.

Likewise, to improve repeatability, a RadarControllerGUI 'ScanMode' (see Chapter 2) was developed to conduct the element pattern measurement, which performs the following actions in sequence:

1. Recenter the array at an azimuth of 0.00° with respect to true north,
2. Configure the analog beamformer controller with weights for generating commutative element patterns, which steps through each element's isolated beam in sequence pulse-to-pulse,
3. Configure the system to generate a continuous pulse train (with a user adjustable T_{PRT}) until interrupted,
4. Start producing pulses using both radar nodes configured to operate with a bistatic topology (user may select between transmit or receive mode),
5. Rotate the array by +540.00° in azimuth, ensuring that all elements have contiguous samples measured for the region surrounding each's boresite position,
6. Once the movement is complete, stop generating pulses, interrupt the ScanMode and thereby end the measurement.

Using this ScanMode to measure element patterns ensured that experiments were always conducted following an identical procedure, which was beneficial to the research process and demonstrates the utility of the software discussed in Chapter 2.

3.1.4 High-Speed Analog Beamformer Controller

The CPPAR's four independently-controllable analog beamformers allowed beam steering in azimuth, unique both between polarizations and between transmit and

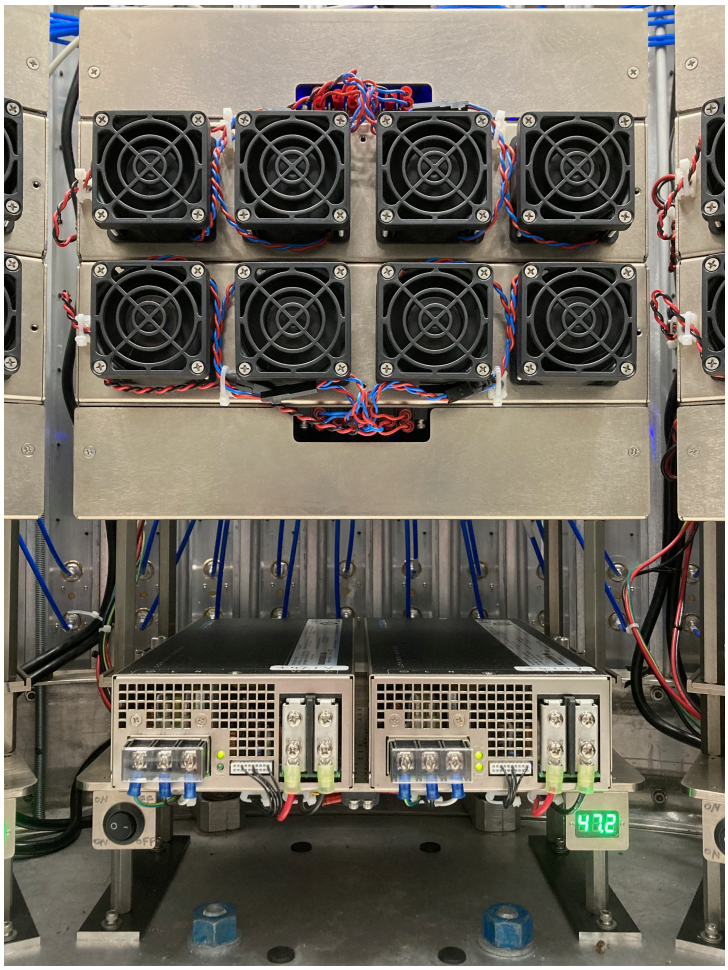


Figure 3.6: One of six submodules from the CPPAR's beamformer, containing driver electronics for eight dual-pol elements.

receive. Each of these beamformers was subdivided into six shared sub-modules (see Fig. 3.6), each containing high-power front end electronics, a power supply, and four parallel phase-amplitude controller (PAC) boards. Each PAC board connected eight elements with a single port, allowing operation as either a power divider (on transmit) or combiner (on receive). In total, this architecture presents 48 antenna columns for beamforming (i.e. half of the full array's 96 antenna columns) yielding a semicircle-shaped array.

Within each PAC, the phase and attenuation acting on each element was

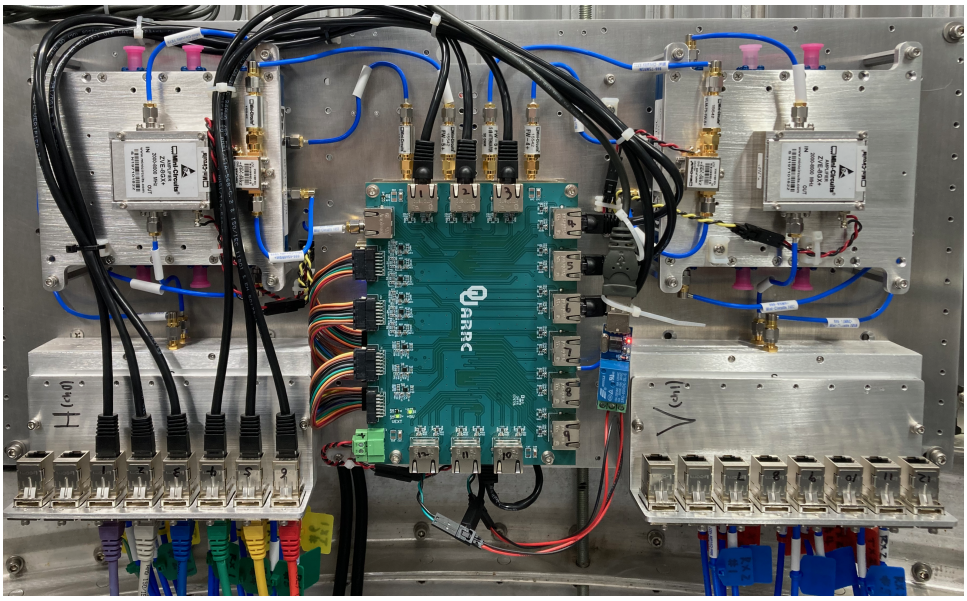


Figure 3.7: CPPAR central electronics board, showing the beamformer controller distribution board (center) and the analog divider / combiner networks (with pre-amplifiers on transmit) leading to each submodule (left and right).

individually controllable by way of digitally-controlled phase shifters and attenuators which provided 360° of phase and 32dB of attenuation control, respectively, with 6 bits of control creating 64 discrete control states (i.e. minimum controllable resolutions of 5.63° and 0.50 dB). In total (including extra control signals), each element's configuration was described by 16 bits of information, suggesting a full-array configuration width of

$$4\,\text{beamformers} \times 48\,\text{elements} \times 16\,\text{bits} = 3072\,\text{total bits} \quad . \tag{3.5}$$

Loading weights onto the array required strategically serializing and shifting these 3072 bits onto the electronics as quickly as possible, as this time delay ultimately determined the minimum T_{PRT} possible when operating the array using flexible pulse-to-pulse beamforming. To place as few barriers as possible on performance, minimizing this latency was thus considered an essential goal during the design

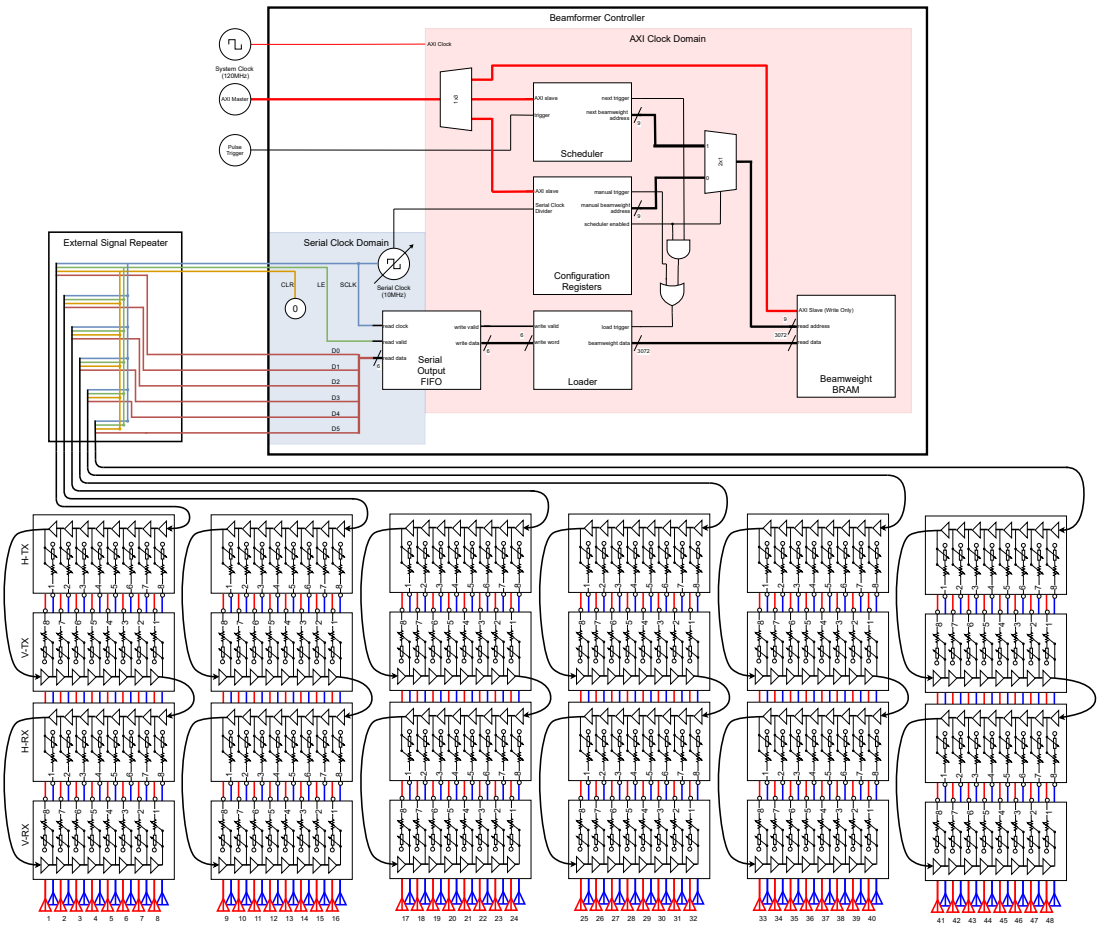


Figure 3.8: Block diagram visualizing CPPAR's beamformer controller design and connectivity.

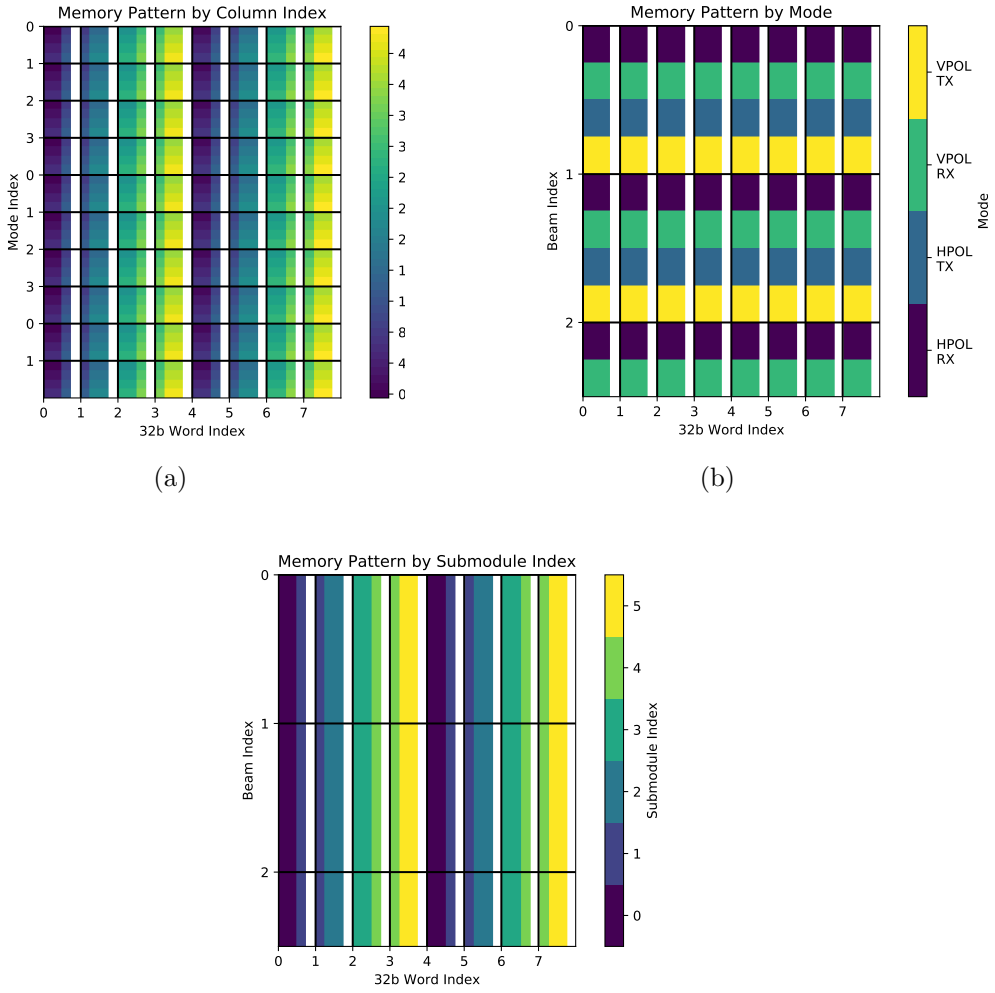
process.

With these considerations in mind, a control topology was designed to enable rapid beamformer configuration via a parallel loading mechanism, with each of the beamformer's six 'submodules' receiving a separate control line. The submodule's four PAC boards—each earmarked to act as either the *HPOL TX*, *HPOL RX*, *VPOL TX*, or *VPOL RX* beamformer for the 8 dual-polarized channels driven by the submodule—were then daisy-chained together to distribute control within each submodule (see Fig. 3.8). Using this design, the latency of a full-array beamformer update is only the time required to load weights onto a single

submodule with its four PAC boards, i.e.

$$\begin{aligned} 4.00\,\text{PAC boards} \times 8.00\,\text{elements} \times 16.00\,\text{bits} = 512.00\,\text{total bits, with} \\ 512.00\,\text{bits} \times \frac{1}{5.00\,\text{MHz}} = 51.20\,\mu\text{s}, \end{aligned} \tag{3.6}$$

which placed a tolerably low constraint on the system's $T_{\texttt{PRT}}$.



(a) (b)

(c)

Figure 3.9: Visualization of the beamformer controller's memory layout.

In the beamformer controller's memory, beams were organized into groups of up to 360, together known as configuration 'sets'. Each of these beams could be

individually indexed and loaded into the array, either manually via control registers or automatically on a pulse-to-pulse basis following a configurable pattern. To reduce the complexity of the FPGA's internal read/write logic, a strategic data format was created which maps from the FPGA's 32-bit memory map to the beamformer's 24-bit word size, growing the block ram's required address space but greatly simplifying the FPGA-logic needed to achieve the six-way parallel loading scheme previously described, since 24 is divisible by 6 but 32 (and indeed, any power of 2) is not. Fig. 3.9 visualizes how one of three distinct analog beamformer features maps to the 24-bit block ram, with each byte colored based on some feature and white regions highlighting 32-bit byte indices not present in the 24-bit memory. Fig. 3.9a shows bytes colored by antenna column index, Fig. 3.9b colored by operational mode, and Fig. 3.9c shows each byte colored by its associated submodule. Additionally, Figs. 3.9b and 3.9c demonstrate the format's compact form-factor, ensuring multiple beams may be stacked together and indexed between with no vacancies, maximizing the efficiency of the storage block ram. In this plot, the y-axis includes dividing lines at the boundary between beams.

	Pulse Group 1				Pulse Group 2			$\cdots$
Pulse Number	1	2	$\cdots$	48	49	50	$\cdots$	$\cdots$
Beam Index	1	2	$\cdots$	48	1	2	$\cdots$	$\cdots$

Table 3.1: Beam selection pattern for interleaved scan.

As part of the control logic for the beamformer, programs may be defined which automatically load beams from memory and sustain them for a configurable number of pulses. To enable element pattern measurements under this architecture, a beamformer program was devised to step through the elements in series (with single-pulse excitations) then repeat the pattern once all elements had been traversed, see Table 3.1. The radar was configured to capture pulses

in groups of 48—matching the number of elements—which guaranteed that as beams were automatically selected, each would always associate with the same index, which allowed the elements' separate patterns to be deinterleaved in post-processing.

Leveraging this beamformer program, element patterns were measured by commanding the CPPAR's bistatic system to continuously transmit / receive while rotating the pedestal at a constant rate. For repeatability during pattern captures, the pedestal always re-positions to a constant azimuthal start position prior to each measurement. Continuously while the pedestal rotates, each element is sampled at an approximately uniform azimuthal interval dependent on both the system's pulse repetition time ($T_{\texttt{PRT}}$) and the pedestal's rotation speed (V_θ). However, the CPPAR's unique shape and the interleaved capture scheme result in each element's samples aligning to different coordinate systems offset from one another. The uniform azimuthal spacing $\theta^{\texttt{EL}}_{\texttt{spacing}} = 3.75°$ between the $N_{\texttt{elements}} = 48$ elements applies a static offset to each, allowing the group to image a 180.00° sector—this feature is noteworthy, and will prove vital in Sections 3.2 and 3.3. Additionally, each element's index within the interleaved pulse group results in a constant time offset from each group's start, which given the pedestal's ongoing rotation leads to a speed-dependent azimuthal skew linked with the pedestal's travel distance during preceding pulses. Together, these parameters control the azimuthal sampling pattern of the data.

3.1.5 GPS Synchronization and Resulting Phase Drift

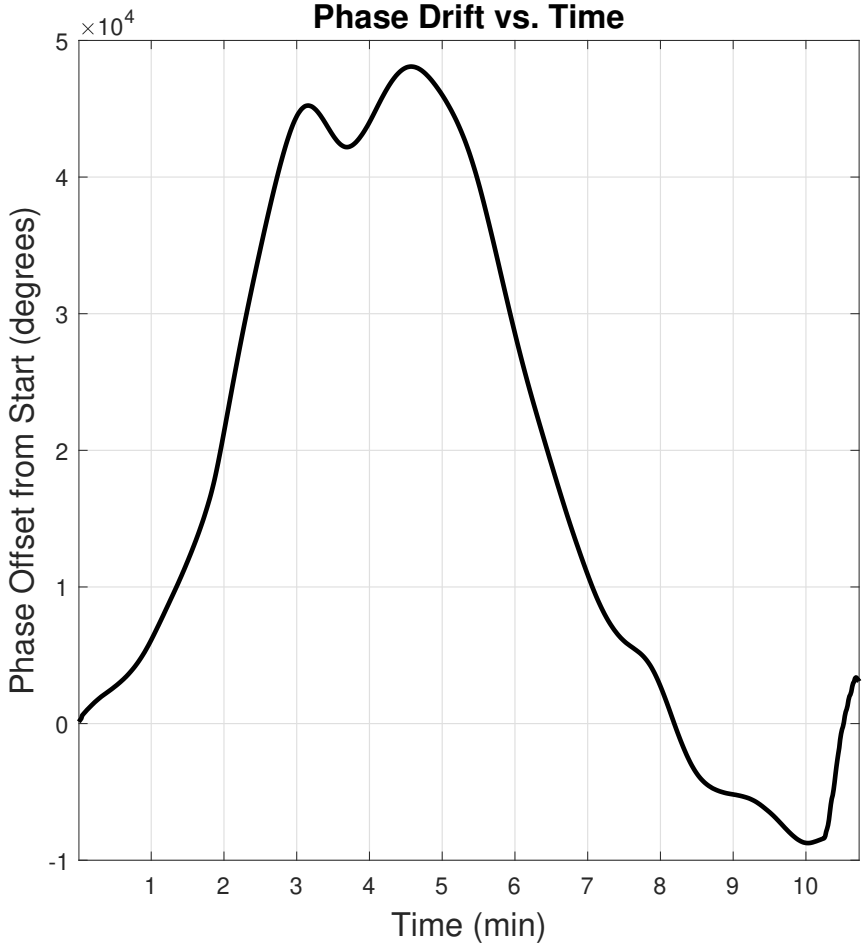


Figure 3.10: Unwrapped phase drift vs. time, estimated from baseband pulses without rotating the array [26]. Reprinted from Herndon and Yeary (2021) © 2021 IEEE

As both the CPPAR and the far-field node are each fully functional radars, the combined system is capable of bistatic measurement in both directions, making it possible to measure both the CPPAR's transmit and receive patterns. If the two nodes' clocking networks were coherent with one another, the procedure outlined in Section 3.1.3 would be sufficient to measure phase information between them without ambiguity. However, in the real system the method used to synchro-

nize the two nodes was subject to instability over time, and resulted in phase drift at baseband which must be managed in order to extract meaningful phase information from the patterns.

Given that the nodes were geographically isolated from one another, direct synchronization was impractical. Instead, the nodes were provided with GPS timebases which generate a one pulse-per-second (PPS) trigger and a 10.00 MHz reference clock. Through experimentation and in line with the GPS's documented capabilities, it was found that the timebase's 1 PPS trigger was precise enough to synchronize the two nodes' pulse envelopes to within a single clock cycle of the digital transceiver's 30.00 MHz baseband sampling rate. Likewise, the 10.00 MHz oscillators act as phase references for each transceiver's clocking network, driving the deterministic logic controlling each radar's operation by way of their on-board field-programmable-gate-arrays (FPGA) while also providing references for the phase-locked loops (PLL) embedded within their transceivers. Although highly disciplined, small frequency differences are inevitable between these two reference clocks which are critically prone to changes over time, as each timebase is subjected to varying environmental conditions. When multiplied to S-band within the PLLs, these effects are amplified and lead to significant frequency differences between the two radar's carriers. At baseband, this frequency difference is observed as rolling phase effect with a constantly shifting velocity. From a stationary reference point, this effect takes the form of additive *phase drift* over time (see Fig. 3.10). Later in Section 3.4, Fig. 3.19 shows aligned phase patterns for the array elements both before and after drift is corrected for in the measured patterns, demonstrating the effect's often dramatic impact on measurement quality.

Several options were considered for managing or mitigating the effects of phase

drift:

1. a hardware solution, wherein a stable clock reference is shared between both nodes by passing it over-air to the farfield and used to directly enforce coherence between the two nodes;

2. a combined hardware / software solution, wherein one of the CPPAR's columns is sacrificed and used to provide a geometrically stationary phase reference for estimating drift in post-processing from the baseband data; and

3. a pure-software solution, wherein the system's phase drift is estimated from the baseband data and subtracted out of the patterns in post-processing.

Option (1) required substantial hardware changes, and was as such considered an option of last resort. Both (2) and (3) would rely on similar post-processing methods to estimate drift from captured baseband data, which would allow progress made on the pure-software solution to be leveraged if required by the combined solution. For this reason and to avoid sacrificing one of the CPPAR's channels, we elected to follow option (3), leaving option (2) as a fallback in the event that our initial efforts failed. Ultimately, a method was successfully developed which carefully considers the array's geometry and its relationship to the environment in order to measure phase drift in post-processing without the need for hardware changes.

3.1.6 Phase Drift Estimation via Integration of Phase Velocity

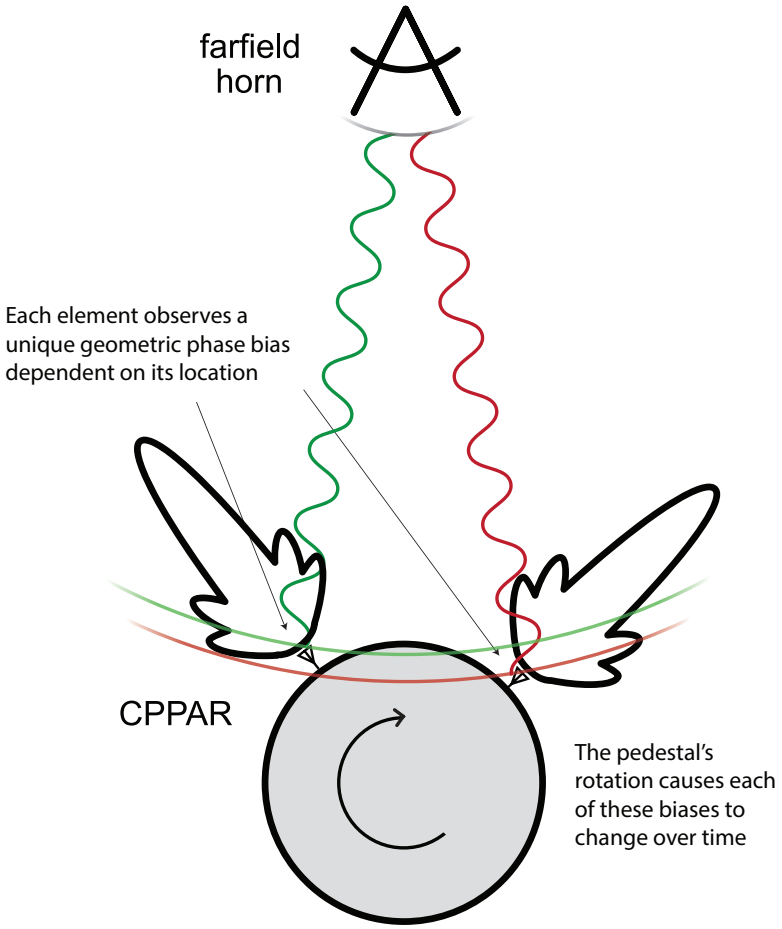


Figure 3.11: Visualization of the phase relationship between array elements in the CPPAR and the farfield horn. The line-of-sight transmission length is substantially different between elements, which creates a time-dependent non-linear phase shift unique to each element as the array rotates [26]. Reprinted from Herndon and Yeary (2021) © 2021 IEEE

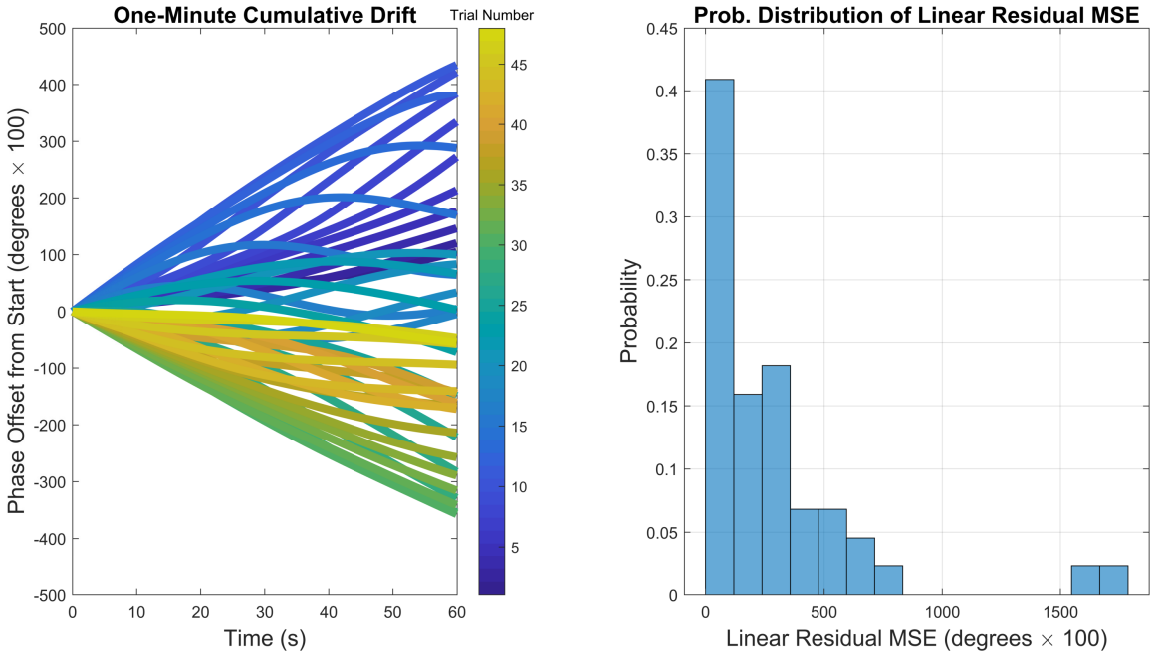


Figure 3.12: Many overlaid stationary drift measurements of approximately the same duration as our pattern measurements [26]. Reprinted from Herndon and Yeary (2021) © 2021 IEEE

Measuring baseband phase drift in the system requires establishing a stable phase reference. In this capacity, a single moment from within the measured patterns may be defined as the phase zero from which all other phases are referenced. Deviations from this reference may then be measured over time to determine the amount of phase drift observed by the system. When the CPPAR is stationary, this method is sufficient to characterize the system's drift (see Fig. 3.10). When in motion, however, time-dependent phase biases caused by the bistatic system's constantly changing measurement geometry complicate phase measurements (see Fig. 3.11).

By re-framing the problem, however, a more general method was developed which defines relative phase in terms of phase velocity, an approach which extends drift measurement to cases where a spatially constant and continuously observable phase reference cannot be assumed such as during rotation when time-

varying geometry-dependent phase effects complicate any attempt to measure drift using simpler techniques. For some arbitrary discrete-time vector of phase measurements ϕ of length k, instantaneous phase velocity may be expressed as the difference relation

$$\begin{aligned} \frac{d\phi}{dt}[n] &= \phi[n+1] - \phi[n] \\ &\text{with } n \in \{0, 1, \cdots, \mathrm{k}-1\} \ . \end{aligned} \tag{3.7}$$

This expression also implies its inverse, which derives the original measurement from the instantaneous phase velocity function via cumulative summation

$$\phi[n] = \sum_{m=0}^{n} \frac{d\phi}{dt}[m] \ . \tag{3.8}$$

To demonstrate this technique, bistatic phase was tracked for approximately ten minutes from a single element while the pedestal remained stationary—Fig. 3.10 shows unwrapped phase vs. time estimated from this data. From a different perspective, Fig. 3.12 shows the same data set with many one minute slices plotted simultaneously, demonstrating the generally unpredictable and often nonlinear behavior of phase drift over such timescales. Of particular importance is the random distribution of these slices and each's markedly nonlinear shape, suggesting a constantly shifting phase velocity. Evaluating these observations further, linear models were fit to each slice and used to score the drift velocity's stability as the mean-squared error of its residuals. Fig. 3.12 shows a histogram of the scores from these trials—note the large average deviation from linearity. This experiment also proved that drift may be practically measured with no other significant time-dependent phase effects present, an important proof-of-concept which justified further efforts.

3.1.7 Critical Challenges

From Section 3.1.2, equations (3.1) and (3.2) establish that calibration terms may be derived for each element in both power and phase, measured as the average bias between each element's pattern and the average of all pattern. Power patterns may be captured trivially via the methods already described. In measured bistatic phase patterns, however, the inter-element differences which our error model relies on were obscured by the effects of two substantial distinct time-dependent effects.

To estimate the relative phase offset between elements, phase measurements from spatially analogous positions (e.g. boresight) must be sampled from each element. As patterns are measured from a single point (i.e. the far-field node), the array must be rotated to observe these sampling points, causing the first time-dependent phase effect observed in the pattern data. Each element's observed phase is thus biased by its unique line-of-sight distance with respect to the far-field node, which constantly changes as the array rotates, as shown in Fig. 3.11. Though this effect substantially alters the phase patterns, it is deterministic and may thus be modeled, making way for post-processing spatial alignment.

Harder to account for, however, was the second time-dependent phase effect observed in the patterns, which resulted from the nonlinear phase drift discussed in Section 3.1.5. Given a 10.00 °/s rotation rate, the array requires 36.00 s to complete one full rotation. During this time, phase drift resulting from mismatches between the two nodes' GPS-disciplined reference clocks applies a constantly changing phase shift. This effect is applied non-uniformly across the array as a result of the interleaved measurement scheme; spatially similar samples are observed at different times for different elements with geometric different phase offsets (see Fig. 3.5) Taken together, these effects complicate any naive attempts to remove the effects of phase drift in post-processing.

3.2 Sampling Geometry

In Section 3.1.3, our method for capturing element patterns using the CPPAR bistatic system was discussed. The technique leverages beam commutation to measure all elements' patterns in a single rotation of the pedestal by interleaving $\mathrm{N}_{\texttt{elements}}$ element excitations into a single scan, repeated until the conclusion of the rotation. The two-dimensional data set generated by this measurement may be usefully represented using two distinct views—a slow time view, wherein mechanical rotation is the primary source of azimuthal separation between samples from a single element, and a fast time view in which electronic beam commutation is the primary source of azimuthal separation between samples from a single interleaved scan. In the literature, the notion of *corner turning* [12, 34, 58], etc., identifies an operation in which data stored as a two-dimensional matrix may be transposed to provide a parallel sampling relationship embedded in the data along its non-primary axis to further processing. For our purposes, this term usefully clarifies the relationship between the fast time and slow time data samples and reinforces the notion that all samples must be collected and organized before our post-processing algorithms may be performed.

When measured, data captured from CPPAR is organized temporally. For some number of pulse groups $\mathrm{N}_{\texttt{samples}}$ and some number of commutative beam positions $\mathrm{N}_{\texttt{beams}}$, patterns are captured as a continuous pulse train of length $\mathrm{N}_{\texttt{samples}} \cdot \mathrm{N}_{\texttt{beams}}$ with a uniform *fast time* spacing of $\mathrm{T}_{\texttt{PRT}}$, with pulse times

$$\begin{aligned} &\mathrm{P}_{t_f}\left[n_p\right] = n_p \cdot \mathrm{T}_{\texttt{PRT}}, \\ &\text{with } n_p \in \left\{0, 1, \cdots, \mathrm{N}_{\texttt{samples}} \cdot \mathrm{N}_{\texttt{beams}} - 1\right\} \quad . \end{aligned} \tag{3.9}$$

Given the system's interleaved measurement scheme, pulses may be remapped

from their 1D index n_p into 2D data structure indices

$$\begin{aligned} n_{\texttt{beam}} &= \texttt{floor}\left(\frac{n_p}{\mathrm{N}_{\texttt{beams}}}\right) \cdot \mathrm{N}_{\texttt{beams}} \\ n_{\texttt{pulse}} &= \texttt{mod}\left(n_p, \mathrm{N}_{\texttt{beams}}\right) \quad . \end{aligned} \tag{3.10}$$

The first dimension of this structure groups pulses by commutative beam position and the second indexes pulses. Leveraging these indexing coordinates, the timing matrix for a given commutative measurement is populated using

$$\mathbf{T}_{\mathrm{N}_{\texttt{beams}} \times \mathrm{N}_{\texttt{samples}}}\left[n_{\texttt{beam}}, n_{\texttt{pulse}}\right] = n_{\texttt{beam}} \cdot \mathrm{N}_{\texttt{beams}} \cdot \mathrm{T}_{\texttt{PRT}} + n_{\texttt{pulse}} \cdot \mathrm{T}_{\texttt{PRT}} \quad . \tag{3.11}$$

When this organizational scheme is applied to the pattern data itself, what results is called the *pattern matrix* and is symbolically represented as $\mathbf{X}_{\mathrm{N}_{\texttt{beams}} \times \mathrm{N}_{\texttt{samples}}}$.

Recall Fig. 3.4, which visualizes the relationship between each element's radial position in the array and the artificial boresight which determines the pedestal's current position. Leveraging the pedestal's rotation speed (V_θ), the *spatial sampling matrix* was defined to compute each pulse's true azimuthal position from the timing matrix. For an element pattern measurement with $\mathrm{N}_{\texttt{elements}} = 48$ commutative beam positions spanning a 180.00° field-of-view, the transformation from temporal to spatial coordinates takes the following form:

$$\boldsymbol{\theta}_{\mathrm{N}_{\texttt{beams}} \times \mathrm{N}_{\texttt{samples}}}\left[n, m\right] = n\frac{\pi}{48} - \left(\frac{\pi}{2} + \frac{\pi}{2 \cdot 48}\right) + \boldsymbol{\theta}_{\texttt{PD}}\left[n, m\right] \tag{3.12}$$

where

$$\boldsymbol{\theta}_{\texttt{PD}}\left[n, m\right] = \mathbf{T}_{\mathrm{N}_{\texttt{beams}} \times \mathrm{N}_{\texttt{samples}}}\left[n, m\right] \cdot \mathrm{V}_\theta \tag{3.13}$$

specifies the array's pointing direction in terms of the pedestal's rotation velocity V_θ and the timing matrix from equation (3.11) for some element n and at sample

point m. Given this theoretical foundation, the CPPAR's geometric constants ($\theta^{\texttt{EL}}_{\texttt{offset}}$ and $\theta^{\texttt{EL}}_{\texttt{offset}}$ as per equations (3.3) and (3.4)) were substituted to yield the following formulation for CPPAR element patterns:

$$\begin{aligned} \boldsymbol{\theta}_{\mathrm{N}_{\texttt{beams}} \times \mathrm{N}_{\texttt{samples}}}[n,m] \; &= \boldsymbol{\theta}_{\texttt{PD}}[n,m] - \theta^{\texttt{VB}}_{\texttt{offset}} + \theta^{\texttt{EL}}_{\texttt{offset}}[n] \\ &\text{with} \\ &\text{beam index } n \in \{0,1,\cdots,\mathrm{N}_{\texttt{elements}}-1\}, \text{ and} \\ &\text{sample index } m \in \{0,1,\cdots,\mathrm{N}_{\texttt{samples}}-1\}. \end{aligned} \tag{3.14}$$

This model computes the azimuthal broadside position for a given element at some time within the scan using the radar's pointing direction, rotation speed, and pulse repetition time.

3.2.1 Spatial Coherence

For most combinations of sampling parameters, the formed azimuthal grid is unique at each point, with no two elements observing (at broadside) identical absolute spatial positions. If desired, sampling parameters may be strategically selected to avoid this non-uniform spacing, resulting in patterns which *as measured* are aligned to a shared coordinate grid. For example, commutative scans were designed with this alignment characteristic in research conducted by Li et. al. [39, 40] to ensure that sets of commutative beams would consistently illuminate meteorological targets while the CPPAR was rotated, making valid statistical comparisons between each beam's measurements possible. Hereafter, patterns which share a common coordinate grid in this way shall be referred to as *spatially coherent*. In terms of the physical array, this characteristic of measured patterns is achieved when the distance the array rotates over the duration of each commutative cycle matches the spatial distance between commutative beam po-

sitions, a condition which when present yields a spatial sampling matrix meeting the following criterion:

$$\left|\boldsymbol{\theta}_{\mathrm{N}_{\texttt{beams}}\times\mathrm{N}_{\texttt{samples}}}\left[n,m\right]-\boldsymbol{\theta}_{\mathrm{N}_{\texttt{beams}}\times\mathrm{N}_{\texttt{samples}}}\left[n,m+1\right]\right|=\theta^{\mathrm{EL}}_{\texttt{spacing}} \tag{3.15}$$

for all $n \in \{0, 1, \cdots, \mathrm{N}_{\texttt{beams}} - 1\}$ over some contiguous span of samples indexed by m, which implies that all commutative channels sample the same azimuthal positions over this span.

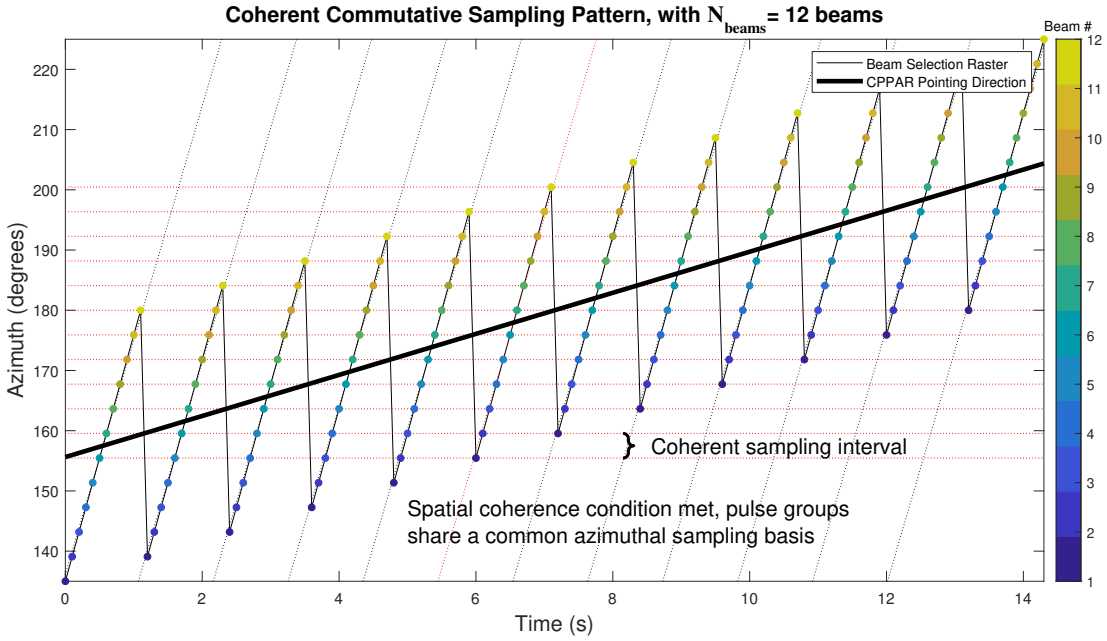


Figure 3.13: Spatially coherent commutative scan [26]. Reprinted from Herndon and Yeary (2021) © 2021 IEEE

As an example, consider a hypothetical commutative beam set with $\mathrm{N}_{\texttt{beams}} = 12$ beam positions spaced apart by $\theta^{\mathrm{EL}}_{\texttt{spacing}} = 3.75^{\circ}$ in azimuth with a target pulse repetition time of $\mathrm{T}_{\texttt{PRT}} = 100.00\,\mathrm{ms}$. The following equation defines the pedestal's azimuthal travel, D_θ, over the duration of each cycle of the commutative scan for

some pedestal rotation velocity V_θ:

$$\mathrm{V}_\theta \cdot \mathrm{T}_{\mathtt{PRT}} \cdot (N-1) = D_\theta. \tag{3.16}$$

When solved for V_θ while letting $D_\theta = \theta^{\mathtt{EL}}_{\mathtt{spacing}}$, an expression for the pedestal rotation speed required to generate *as measured* spatially coherent patterns follows:

$$\mathrm{V}_\theta{}' = \frac{\theta^{\mathtt{EL}}_{\mathtt{spacing}}}{(N-1) \cdot \mathrm{T}_{\mathtt{PRT}}} = 3.41\,^\circ/\mathrm{s} \tag{3.17}$$

In this scenario, spatial coherence is achievable as both $\mathrm{T}_{\mathtt{PRT}}$ and the required V_θ are valid parameters for the system. In Fig. 3.13, the azimuth vs. time sampling pattern of this commutative beam scan is visualized, with samples from all commutative beam positions falling onto the same set of points in azimuth. When configured with these parameters, spatial coherence is achieved but time resolution is limited by $\mathrm{T}_{\mathtt{PRT}}$.

Measurements requiring precise time resolution must navigate the parameter-space created by $\mathrm{T}_{\mathtt{PRT}}$ and the inversely-related V_θ, with decreases in $\mathrm{T}_{\mathtt{PRT}}$ requiring a corresponding increase in V_θ to maintain the spatial coherence condition. High time resolution is an essential requirement if phase patterns are desired, as the effects of GPS phase drift must be measured and accounted for to yield meaningful patterns. Through experimentation, it was found that sampling the system's phase state at around 1000.00 Hz (corresponding to $\mathrm{T}_{\mathtt{PRT}} = 1000.00\,\mu\mathrm{s}$ and a per-element sample rate of around 21.00 Hz) provides sufficient information to estimate drift effects during element pattern measurement. With the system's time resolution thus bounded, meeting the coherence *as measured* condition would require rotating the array at a speed of $79.20\,^\circ/\mathrm{s}$, far faster than the pedestal's maximum rotation speed of $13.00\,^\circ/\mathrm{s}$, making this criteria impos-

sible to meet in this context. Knowing this, the coherent sampling condition was relaxed for element pattern measurements, resulting in patterns which are not spatially coherent. Instead, samples from the pattern matrix are aligned to a skewed affine grid, resulting in both spatial and temporal ambiguity. In Fig. 3.14, a similar grid is shown representing the sampling pattern for a scan with $N_{\texttt{beams}} = 12$ commutative beam positions.

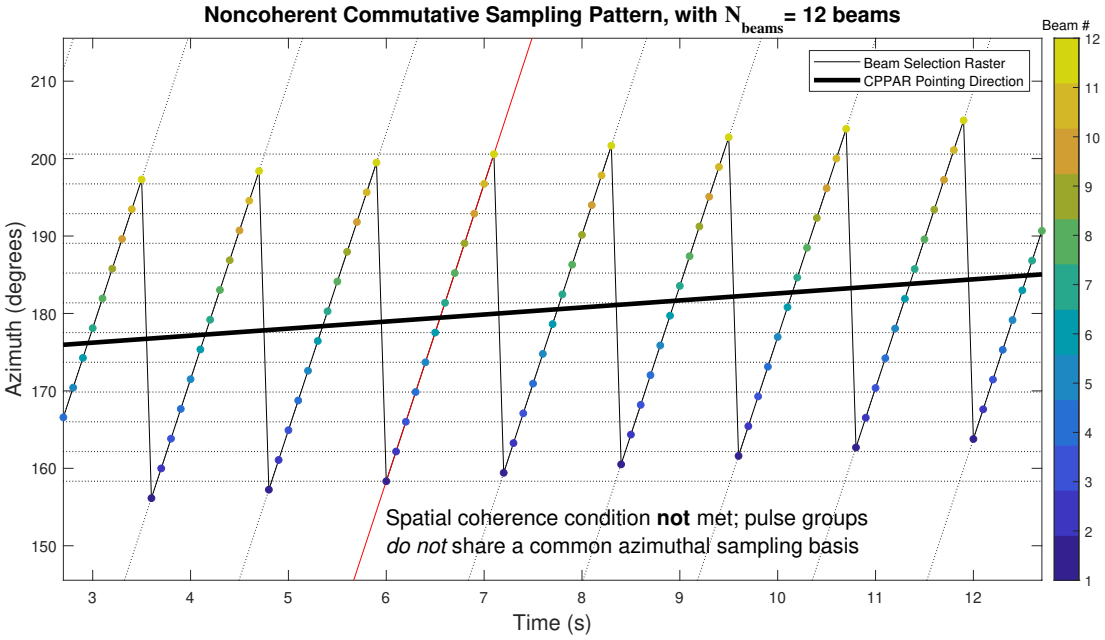


Figure 3.14: Non-coherent commutative spatial sampling pattern [26]. Reprinted from Herndon and Yeary (2021) © 2021 IEEE

3.2.2 Enforced Coherence via Pattern Matrix Re-Sampling

As discussed in Section 3.2.1, our method for measuring element patterns generates a sampling geometry ambiguous in both space and time, an artifact which must be resolved before meaningful comparisons are possible between patterns. As the array's sampling geometry is known and well-defined (see Section 3.1.3), the patterns may be made coherent along one of either data dimensions by inter-

polating samples from their known spatial / temporal position onto a new unified coordinate basis which better represents spatial relationships known to exist in the data, but which are not directly embedded in the sampling. Algorithm 1 describes our procedure for applying this transformation. When a pattern matrix

Algorithm 1: Procedure for spatially aligning a given pattern matrix along its primary dimension.

Data: Pattern matrix $\mathbf{X}_{\mathrm{N}_{\texttt{beams}} \times \mathrm{N}_{\texttt{samples}}}$ aligned to sampling coordinates $\boldsymbol{\theta}_{\mathrm{N}_{\texttt{beams}} \times \mathrm{N}_{\texttt{samples}}}$ as-measured, both of size $\mathrm{N}_{\texttt{beams}} \times \mathrm{N}_{\texttt{samples}}$

Let: n_w act as a window term, slicing columns with index p from $\mathbf{X}_{\mathrm{N}_{\texttt{beams}} \times \mathrm{N}_{\texttt{samples}}}$, with $p \in [\mathrm{n}_w, \mathrm{N}_{\texttt{samples}} - \mathrm{n}_w]$. This constrains the alignment procedure such that only the portion of azimuth space bounded by points sampled by *all* commutative beam positions is considered, allowing well-formed interpolation,

Let: $\hat{\mathrm{N}}_{\texttt{samples}} = \mathrm{N}_{\texttt{samples}} - 2 \cdot \mathrm{n}_w$ define the length of this spatially coherent subset, and lastly

Let: $\hat{\theta}_{1 \times \hat{\mathrm{N}}_{\texttt{samples}}}$ represent some desired common coordinate basis.

Procedure:

for $n \in \{1, 2, \cdots, \mathrm{N}_{\texttt{beams}}\}$ **do**

  Estimate monotonic piece-wise cubic interpolator (pchip) function $f(x)$, for

$$\mathbf{X}_{n,i_2} \text{ aligned to } \boldsymbol{\theta}_{\mathrm{N}_{\texttt{beams}} \times \mathrm{N}_{\texttt{samples}}}[n, i_2],$$
$$\text{with:}$$
$$i_2 \in \{\mathrm{n}_w, \mathrm{n}_w + 1, \cdots, \mathrm{N}_{\texttt{samples}} - \mathrm{n}_w\}$$

  for $i_1 \in \{1, 2, \cdots, \hat{\mathrm{N}}_{\texttt{samples}}\}$ **do**

    Compute $\mathbf{Y}_{n,i_1} = f\left(\hat{\theta}_{i_1}\right)$;

Result: Coherent pattern matrix $\mathbf{Y}_{\mathrm{N}_{\texttt{beams}} \times \hat{\mathrm{N}}_{\texttt{samples}}}$ aligned to common coordinate vector $\hat{\theta}_{1 \times \hat{\mathrm{N}}_{\texttt{samples}}}$

is processed in this way, point-to-point comparisons are made valid along a single data dimension with minimal information loss, allowing vector processing to act as a convenient and statistically valid shortcut for measuring relative differences between slices from the coherent dimension. Next, this idea as applied to both pattern dimensions will be described, along with implications of and use cases

for these two unique interpretations of the data.

Element Boresight Alignment

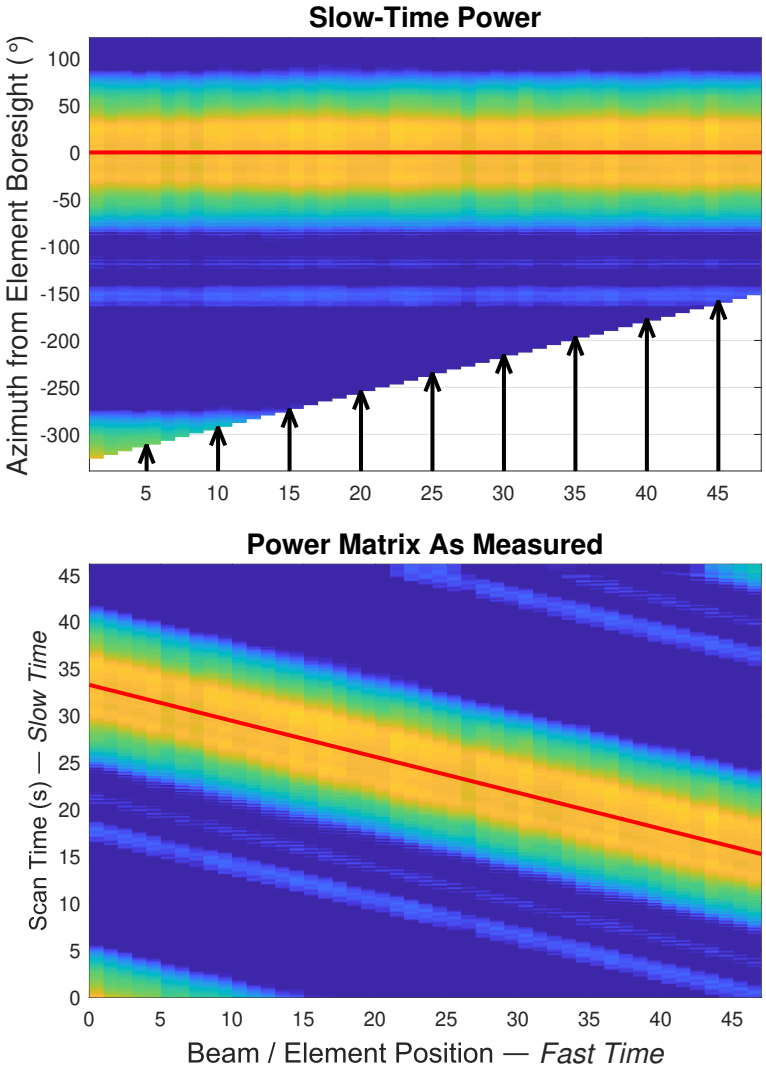


Figure 3.15: A visualization of the transformation from *as measured* geometry onto the slow-time spatially coherent basis [26]. Reprinted from Herndon and Yeary (2021) © 2021 IEEE

In physical terms, slices along the slow-time dimension of a given pattern matrix represent samplings of a single commutative beam's power and phase as it was rotated through the farfield calibration horn's beam. Applied to our pattern

model in combination with the known azimuthal path traced by the element's boresight position, these slices represent realizations of the array's power and phase element patterns $P_n(\theta)$ and $\phi_n(\theta)$. Element patterns form the basis of our power and phase error models for the array [equations (3.1) and (3.2)], which form the theoretical basis for our array calibration method. Therefore, accurate estimates of these patterns are critical to improving the precision of calibration weights. Implicit in this model was the assumption that the variable term 'θ' represents boresight-relative azimuth for each element 'n', which given our new terminology demands that the model's realizations $P_n(\theta)$ and $\phi_n(\theta)$ are *spatially coherent* at all points of measurement.

In Section 3.2.1, it was shown that our method for measuring element patterns necessarily yields non-coherent spatial sampling, and thus produces patterns which must be aligned along the slow-time axis before calibration weights can be estimated. Recall that equation (3.14) defines the sampling matrix, with the primary dimension indexing between elements and the secondary dimension indexing by pulse number. To derive element patterns from these pattern matrices, Algorithm 1 was applied to cohere between slices from the element indexing dimension. See Fig. 3.15 for a two-dimensional visualization of this operation. The resultant pattern matrix demonstrates spatial coherence between slices from the primary dimension, each representing samplings of each element's power and phase with respect to azimuth. Once coherent, these slices were made meaningfully applicable to our array error models and thus allowed for the estimation of calibration weights through basic vector operations, discussed later in Section 3.4.

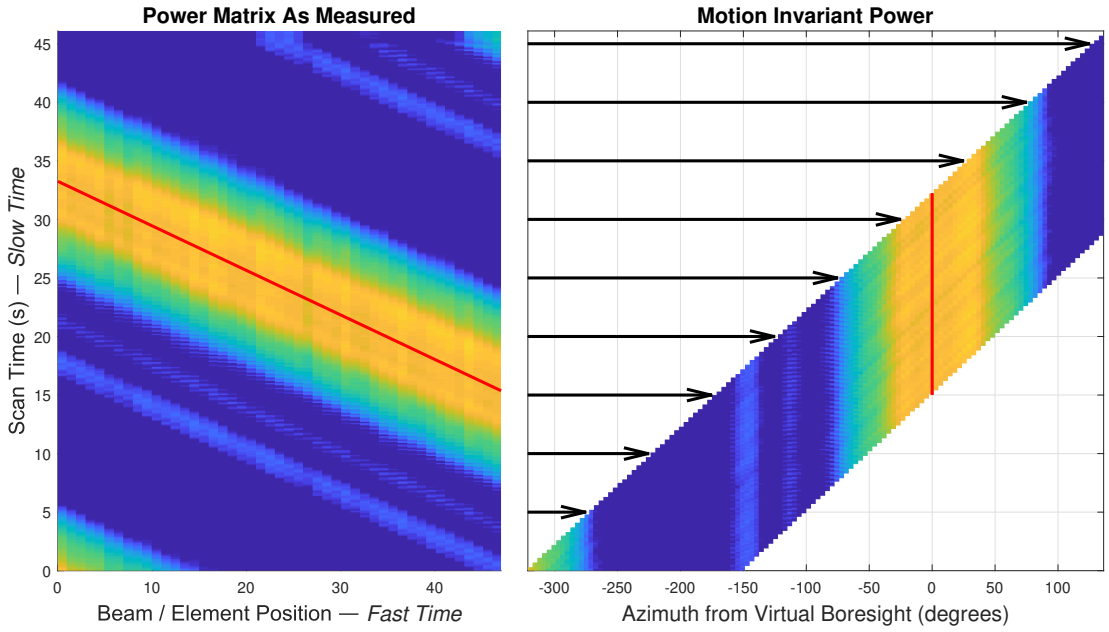


Figure 3.16: A visualization of the transformation from *as measured* geometry onto the fast-time spatially coherent basis, i.e. the *motion invariant* basis [26]. Reprinted from Herndon and Yeary (2021) © 2021 IEEE

Motion-Invariant Alignment

Accurately measuring phase drift within the bistatic system requires establishing a geometrically stable phase reference. When the pedestal is stationary, the measurement's geometry is constant throughout the scan—as a result, the *phase zero point* must be defined as a single arbitrarily chosen sample time within, with all other samples in the pattern matrix defining phase as an offset from this zero point. On the contrary, this stable geometry assumption is no longer valid when the pedestal rotates and thus requires establishing a time-varying (or alternately, geometry dependent) phase reference. Initially, this moving phase reference was approximated by tracking the broadside phase of the *virtual* element pattern created by the elements' interaction with the farfield node. This method achieved limited accuracy, however, and thus a more general approach was sought.

Although the spatially coherent element patterns described in Section 3.2.2 are

perhaps the most intuitive interpretation of the pattern matrix, the operation's analog on the corner-turned pattern matrix exposes a similarly useful though fundamentally unique view of the data. This operation results in a pattern matrix made coherent along the original measured pattern's *primary* dimension, thereby enforcing spatial coherence between fast-time slices. When each element's spatial offset from boresight is accounted for, the resulting matrix exposes the *inverse pattern* which images the calibration horns' antenna patterns. To demonstrate, Fig. 3.16 shows this transformation as applied to measured power patterns.

The motion-invariant transformation fundamentally re-imagines the pattern matrix by generating what will hereby be called a *virtual* element pattern at each time step over the course of the scan. These virtual patterns obfuscate the spatial sampling boundaries previously embedded in the patterns by the measurement process, trading them instead for temporal coherence between fast-time slices. In this new view, geometry-dependent phase effects are aligned along the data's secondary dimension, and as such may be ignored when operating along that dimension. Likewise, time-dependent phase effects unrelated to the pedestal's motion are made temporally coherent between spatial slices from the virtual pattern. Row-wise operations (slices in time) provide a simple method for estimating system-level phase state. Following this general philosophy, a two-stage algorithm was developed to estimate phase drift during commutative pattern measurements.

3.3 Phase Drift Estimation

In the unprocessed pattern matrix, two primary effects obfuscate the common mode phase velocity: the pedestal's motion, which adds a position-dependent nonlinear bias to each element's observed phase; and the measurement strategy

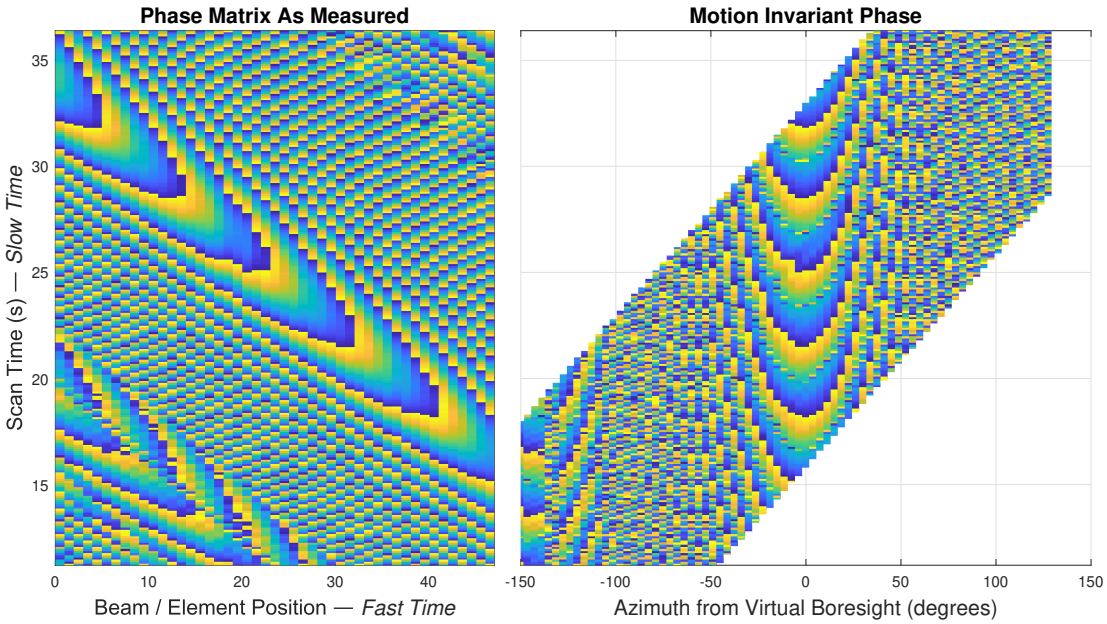


Figure 3.17: Wrapped phase patterns interpolated onto the motion invariant coordinate basis [26]. Reprinted from Herndon and Yeary (2021) © 2021 IEEE

itself, which interleaves excitations of the array's many elements to capture their patterns in a single rotation, resulting in each sampling the common mode phase at offset times from one another. In our method, it was useful to consider how the general concept of phase velocity as defined in equation (3.7) could be usefully applied to the CPPAR system's bistatic architecture. The *common mode* phase velocity was defined as the *true* rate of change of GPS phase drift as observed at any time by all elements in the array, as distinct from the rate of change observed by a single element at a single sample time subject to geometry effects. GPS phase drift was by far the most prominent effect contributing to the system's variable common mode phase velocity, thus measuring this feature was considered sufficient for estimating phase drift.

3.3.1 Common Mode Estimation from Motion-Invariant Phase

In Section 3.2.2, our method for enforcing spatial coherence between fast-time slices in the pattern matrix is discussed. In the left panel of Fig. 3.17, the unprocessed pattern matrix demonstrates the effects of the position-dependent geometric phase bias while the array was constantly rotating—the virtual pattern representing the image of the farfield pattern as observed by the array was encoded diagonally in the pattern matrix. The right panel in Fig. 3.17 demonstrates the utility of the motion-invariant phase. In this view, the farfield pattern's virtual pattern was made coherent along the time dimension, making way for the coherent estimation of phase drift velocity.

To measure this effect, an algorithm was developed following the method outlined in Section 3.1.6 which estimates the system's instantaneous phase velocity from the motion-invariant pattern matrix over the duration of the scan, then computes the bias vector via integration of the phase velocity. Using the motion-invariant power matrix as a reference, a threshold was applied to the phase matrix to exclude samples below a configurable SNR. Next, the discrete-time difference function was applied along the time dimension to estimate the phase velocity at each virtual pattern slice. Since many virtual samples are contained within each difference pair, many parallel estimates of the velocity are produced. These estimates are quality controlled for statistical significance via an iterative algorithm which will be excluded from this discussion for brevity, but ultimately results in an estimation of the common mode velocity. Applying discrete integration to this velocity vector yielded an estimate of the phase drift vector over time. Finally, Savitzky-Golay filtering (a polynomial smoothing algorithm defined as a generalization of a moving average filter [3, 7, 11, 45, 52]) was employed to smooth the

resultant time series. The final drift estimate from this stage is then subtracted from the phase pattern to produce a partial result, which is then further refined in subsequent stages of drift correction.

3.3.2 Estimation of Residual Error

Following from our phase error model as defined in equation (3.2), it was assumed that the array's phase biases have a constant distribution with respect to their shared mean. For a given non-coherent pattern matrix with no phase drift, sampling phase from each element's boresight and subtracting away their shared mean will produce an estimate of this bias distribution. In the partially-corrected element pattern matrix, this relationship will still be present but will be obfuscated by the residual drift, which applies a unique bias to each element's phase as a result of their different measurement times. Likewise, if we assume drift but no array errors, an estimate of the drift may be found by tracking how the boresight phase changes over the course of the pattern measurement. Using only a single spatial sampling point only produces as many samples as there are elements, however, which limits the temporal resolution of the drift measurement. To increase the accuracy of this measurement, additional spatial samples may be taken then averaged together at each time step in ensemble. When these methods are combined and when drift in the pattern matrix is sufficiently small, the resulting algorithm is able to effectively disambiguate the two effects and provide a continuous picture of the phase state of the system across time.

3.3.3 Qualitative Verification of Drift Estimates

When the results from the algorithms outlined in Sections 3.3.2 and 3.3.2 are combined, an accurate approximation of phase drift is produced and may thus be

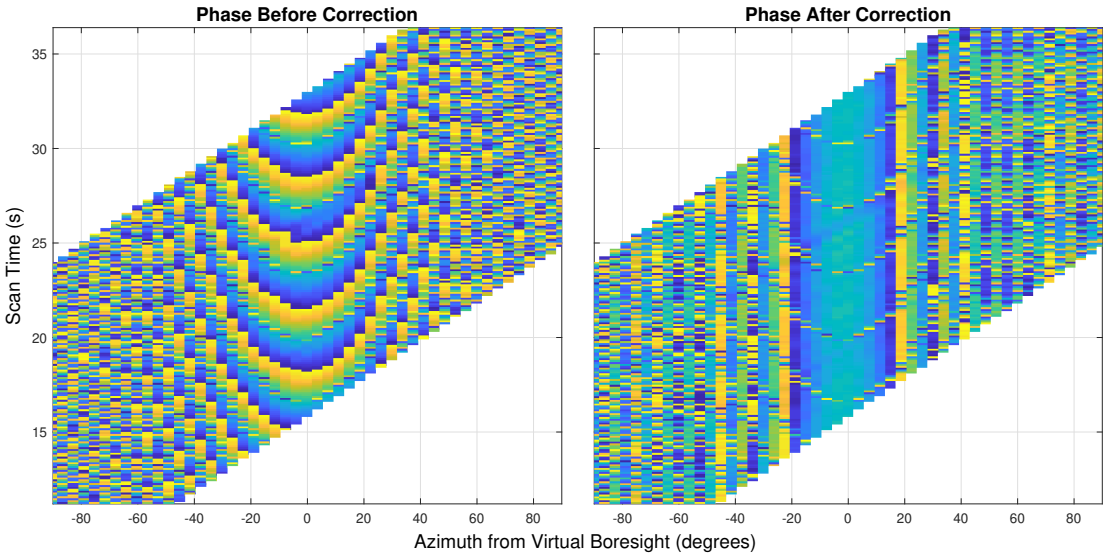


Figure 3.18: Motion-invariant phase matrix before and after drift correction [26]. Reprinted from Herndon and Yeary (2021) © 2021 IEEE

subtracted out of the pattern matrix. Once estimated, it was essential that its accuracy be verifiable in some way. In addition to its utility in reducing rotation-induced phase effects, the motion-invariant matrix provides a high-level visual which proves useful for determining the effectiveness of drift correction. Since our method does not directly observe phase drift, it was essential to develop stable metrics for validating the patterns such that false or inaccurate estimates could be identified more easily. For this purpose, several qualitative heuristics were developed for analyzing the accuracy of drift correction:

- In an ideal pattern measured from a perfectly calibrated array (i.e. phase biases are zero), the image of the farfield calibration horn's pattern on the array will have a constant shape for the duration of the scan.

- As the virtual pattern is formed as a reinterpretation of samples captured from all elements skewed along a diagonal, each element's phase error will be encoded along the diagonal rather than remaining coherent with the virtual boresight's sampling geometry. See Fig. 3.18.

- Drift corrected patterns captured from a calibrated array will thus minimize phase variance for any sufficiently high-SNR slices along the time dimension (e.g. at virtual boresight). See also Fig. 3.18.

3.4 Final Pattern Matrix Correction and Estimation of Element Bias Terms

With a theoretical framework and corresponding methods thus established, we will now discuss how these ideas were combined to form a coherent calibration strategy for the system.

Pattern Matrix Drift Correction

Before any processing can begin, the measured pattern matrix must be drift-corrected. Once established that the estimated phase drift is valid, the resultant time series is simply subtracted from the measured phase pattern matrix along its corresponding time dimension to yield its drift-corrected form.

Boresight Alignment of Element Patterns

Leveraging our sampling model and our method for aligning the elements' physical boresights, the drift-corrected pattern matrix was cohered along the slow-time dimension. Once aligned, the resultant spatially coherent pattern matrix suitably maps to our error model as defined in Section 3.1.2, and thus provides an ample basis for estimating calibration weights. Fig. 3.19 visualizes wrapped phase

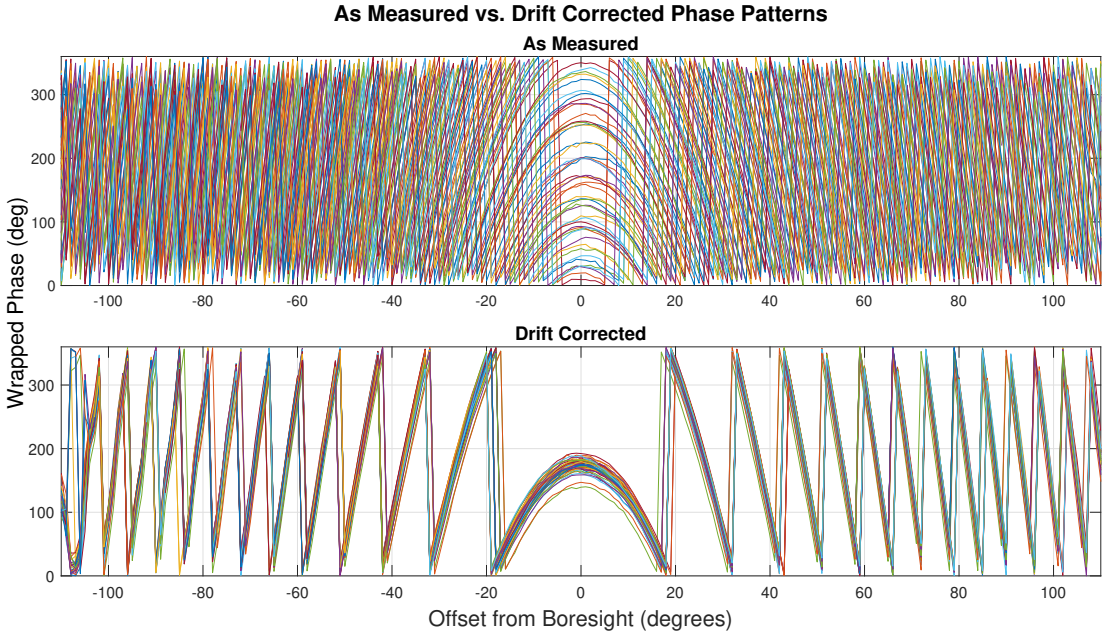


Figure 3.19: Phase patterns for each of the CPPAR's 48 (uncalibrated) elements before and after drift correction [26]. Reprinted from Herndon and Yeary (2021) © 2021 IEEE

patterns captured from CPPAR (with no calibration weights applied) before and after drift correction, demonstrating our method's effectiveness as well as the results of our boresight alignment process.

Bias Estimation from Boresight-Aligned Patterns

Following directly from the calibration model defined in Section 3.1.2, element patterns were leveraged to compute phase and power bias terms. As an initial quality control measure, a subset of the main pattern matrix is formed. Generally, some neighborhood around boresight is selected as this region has the highest SNR and thus provides the least noisy data for bias estimation. Next, for each spatial sampling point in this region, the element-wise mean is computed for the power and phase patterns separately, forming estimates of the ideal power and

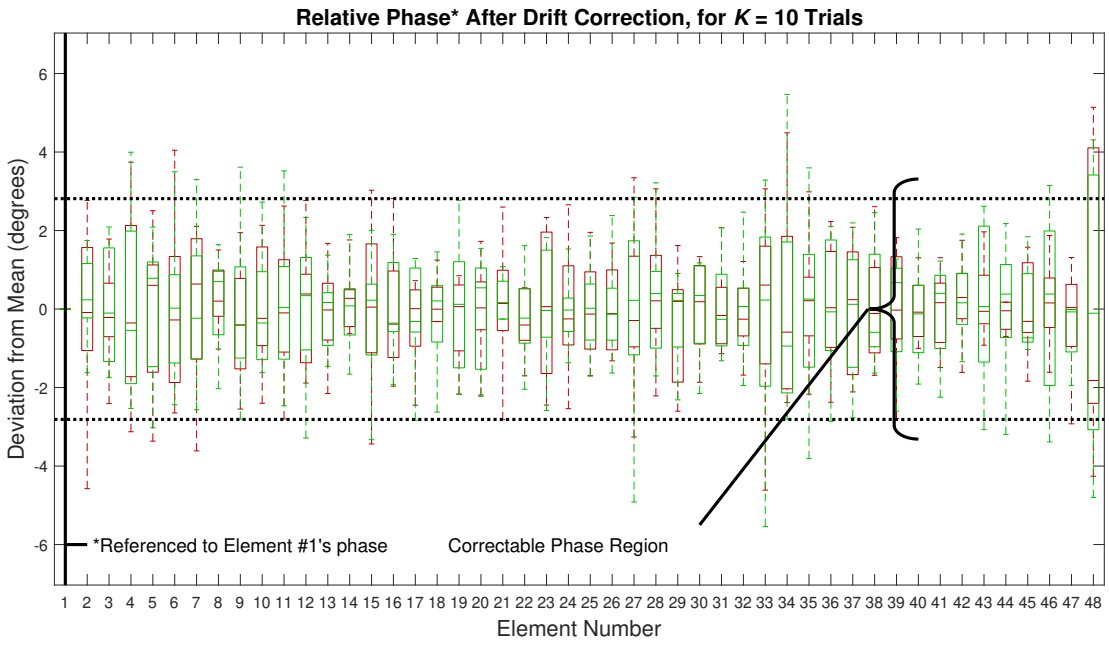


Figure 3.20: Distributions of each element's phase calibration coefficients for $K = 10$ trials [26]. Reprinted from Herndon and Yeary (2021) © 2021 IEEE

phase patterns $|\mathrm{A}(\theta)|$ and $\angle\mathrm{A}(\theta)$. These estimates are then subtracted from each element's power and phase patterns to compute residual vectors, which are then averaged to form estimates of the power and phase error terms from equations (3.1) and (3.2). This procedure is summarized in Algorithm 2.

When applied to measured element patterns, Algorithm 2 is capable of estimating each element's bias terms with sufficient accuracy to calibrate the array to within the resolution of the array's phase shifters and attenuators. To demonstrate the method's effectiveness, $K = 10$ separate element pattern measurements were made for statistical comparison. Each trial's pattern matrix was processed to produce power and phase bias estimates from each trial, then analyzed for consistency. In Fig. 3.20, the distributions of the estimated phase bias terms for each of the CPPAR's active transmit elements (in horizontal and vertical polarization) are represented as a box plot. On average, our method accurately approximates element bias values. The larger uncertainty at element #47, however, reveals

Algorithm 2: Procedure for estimating power or phase biases from the spatially aligned pattern matrix.

Data: $\hat{\mathbf{P}}_{\mathrm{N_{beams}} \times \hat{\mathrm{N}}_{\mathtt{samples}}}$, a subset taken from the $\mathrm{N}_{\mathtt{samples}}$ spatially coherent samples yielded during alignment of length $\hat{\mathrm{N}}_{\mathtt{samples}}$, for each $\mathrm{N}_{\mathtt{beams}} = 48$ array elements.

Let: $\bar{\mathbf{P}}_{1 \times \hat{\mathrm{N}}_{\mathtt{samples}}}$, estimation of the ideal pattern derived for each point $n \in \{1, 2, \cdots, \hat{\mathrm{N}}_{\mathtt{samples}}\}$.

Procedure:

for $i_1 \in \{1, 2, \cdots, \hat{\mathrm{N}}_{\mathtt{samples}}\}$ **do**

$\quad \bar{\mathbf{P}}\left[1, i_1\right] = \mathbf{E}\left[\hat{\mathbf{P}}\left[:, i_1\right]\right]$

for $i_2 \in \{1, 2, \cdots, \mathrm{N}_{\mathtt{beams}}\}$ **do**

$\quad \bar{\mathbf{r}}\left[i_2, 1\right] = \mathbf{E}\left[\hat{\mathbf{P}}\left[i_2, :\right] - \bar{\mathbf{P}}\right]$

Result: Mean residual values for each element collected into the vector $\bar{\mathbf{r}}_{\mathrm{N_{beams}} \times 1}$, representing average bias referenced to the estimated ideal pattern.

one weakness of our method: phase estimates for elements adjacent to the inactive array sections are formed from fewer valid samples compared to elements near the center of the active section. As a result, the phase drift estimate from these points in time have a higher variance and in turn leads to less accurate bias estimates at the effected elements. Regardless, our method on average was sufficient to calibrate the array, making way for successful commutative weather measurements.

3.5 Closing Remarks

In this chapter, a procedure for measuring element patterns for the Cylindrical Polarimetric Phased Array Radar via an external GPS-synchronized calibration horn in the array's farfield was discussed. Although this architecture guaranteed stable bistatic pulse envelope measurements, phase was complicated by a time-

varying frequency offset between the two transceivers' local oscillators which introduced an arbitrary phase drift observable in the baseband IQ data—concealing the relative phase differences between elements in the measured patterns.

To manage this effect, a measurement scheme was designed which exploits the array's cylindrical geometry and rapid beam-switching ability to interleave pattern measurements for all elements into a single scan, ensuring a subset of the elements are always observable from the receiver and thus producing full-array patterns with trackable phase from a single rotation of the CPPAR's mechanical pedestal. Quantifying this procedure's sampling geometry enabled accurate spatial alignment between element patterns in post-processing. Likewise, a reinterpretation of this data along its non-primary axis yielded the so-call *motion-invariant* view of the inverse pattern (i.e. the measurement horn), which allowed phase drift to be measured irrespective of the CPPAR's geometry or motion. Finally, the estimated drift was subtracted from the patterns which were then spatially aligned to yield the final element patterns. With these drift-corrected and spatially coherent patterns, each element's phase and power error terms were estimated and ultimately used to derive calibration weights for the array.

Within the wider context of this thesis, the methods described in this chapter represent case studies in array calibration within a practical system. The simple error model used to derive calibration weights serves as one method for calibrating an array wherein phase and power output differences between elements are distributed relative to some average value, behavior which is present in the CPPAR's analog beamformer network. Likewise, the technique used to measure bistatic phase drift represents one method for resolving phased relationships in baseband data captured from a phased array system subject to time-varying phase, where no stable phase reference is resolvable. Although the specific method discussed

is principally tied to the cylindrical geometry of the CPPAR, the crux of the method—tracking phase drift over time from a moving platform via exhaustive accounting of an array's sampling geometry—holds implications for future efforts with bistatic systems, and therefore warrants discussion.

In the following chapter, another form of system calibration is discussed, which leverages digital techniques prior to RF synthesis to linearize signal gain through high-power amplifiers, correcting for distortion incurred by nonlinearities therein. When applied, this form of calibration—known as digital predistortion—holds implications for a variety of system-level performance metrics, and is thus critical to the success of next-generation phased array systems.

Chapter 4

Real-Time Digital Predistortion for Optimized Performance in Phased Array Systems

Working towards the goal of all-digital phased arrays, next-generation all-digital phased array weather radars in development at the University of Oklahoma's Advanced Radar Research Center (ARRC) will rely on many distinct HPAs—one for each element—working together in an array environment [36, 46]. To maximize the array's power output while still allowing for the flexibility offered by DEE arrays, it is essential that all elements contained within these arrays be capable of operating in saturation without incurring the distortion typically experienced therein. As one of a series of steps for calibrating these arrays, digital predistortion using the memory-polynomial model has been targeted as a candidate for maximizing each element's usable power while minimizing spectral spreading and achieving desirable output linearity during operation.

The literature explores methods for training these models on isolated amplifiers [17], but focuses primarily on measurements captured directly from amplifiers within a bench top environment. In practical phased array systems, the question of how to capture training data from each of an array's elements is vitally important if in-field calibration is desired but where switched RF feedback after the distortion source is not available. To aid in exploring one possible method

for capturing training data in the field, a research and development platform was developed to enable the observation of signals heavily distorted by a high-power amplifier [25]. This platform was configured to transmit both a direct, post-distortion loopback signal (to act as a ground truth) in parallel with an experimental radiative measurement path split from the same source and observed via mutual coupling. In a series of tests, this platform was used to capture baseband IQ from these two measurement paths for both training memory polynomial models and providing useful analytical datasets. Once adequate performance was achieved in simulation, the platform was leveraged to verify predistortion's effectiveness by applying the memory polynomial operation to input waveforms offline, then using these as arbitrary waveforms to drive the system. In addition to providing a platform for validating early-stage predistortion, these experiment provided a glimpse of the challenges which must be overcome when calibrating systems in the field.

Once these basic methods were proved within the testbench system, a high-performance FPGA implementation of the memory polynomial model was developed for applying predistortion to streams of IQ data [24]. Given the targeted flexibility of the digital radar systems the component was developed for, an ideal module would apply predistortion online, in real-time and with minimal buffering, providing the greatest possible waveform diversity (i.e. supporting pulse-to-pulse waveform changes) and avoiding impacts to the system's maximum pulse-repetition-frequency (PRF). To meet these requirements, a custom solution was developed using IntelFPGA's high-level synthesis tool which achieved optimal throughput and in turn minimized system-level constraints for these next generation systems. In the final section of this chapter, this implementation of the memory polynomial predistortion model is described and demonstrated in

both simulation and hardware tests.

4.1 Digital Predistortion Theory

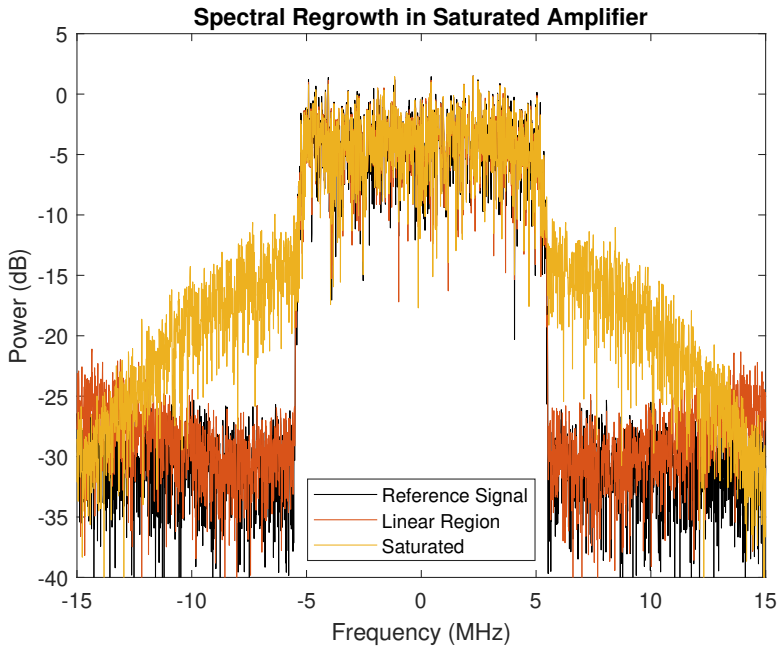


Figure 4.1: Spectral regrowth, seen as the broadened spectrum of a saturated amplifier, as based on our lab's measurements [25]. Reprinted from Herndon, Yeary, and Palmer (2020) © 2020 IEEE

Phased arrays require highly-calibrated elements coordinating together to function as intended. In DEE system specifically, each digital element is paired with an analog front-end containing, among other things, a high-power amplifier. These amplifiers distort signals when operating in saturation, resulting in non-linear gain and spectral regrowth of the form seen in Figure 4.1, unintentionally increasing the effective transmit bandwidth and resulting in lower efficiency as power spreads to out-of-band regions.

Much of this distortion may be summarized as a combination of two effects: amplitude-amplitude (AA) distortion, which takes the form of a non-linear rela-

tionship between input and output power; and frequency-amplitude (FA) distortion, which appears as a frequency dependency in the system's gain. It is well understood that FA distortion may be aptly modeled as a 'memory effect', a result of an amplifier's tendency to 'remember' (i.e. be affected by) previous states of the amplifier as a result of its finite bandwidth [42]. If these distortion effects are not corrected for, designers must either contend with degraded signal quality when operating the radar at or near its peak power or they must ensure that a given system operates its amplifiers close to or fully within their linear regions, limiting the maximum output power achievable by the system and resulting in reduced efficiency. To gain access to an amplifier's full output capacity without incurring unacceptable levels of distortion, amplifier distortion must either be corrected for after-the-fact, or it must be preempted in some way. Predistortion—a blanket term identifying methods of transforming signals prior to their entering some distortion source in an effort to improve the post-distortion characteristics of the signal—attempts to accomplish the latter.

In theory, predistortion requires finding some model of an amplifier's inverse behavior—that is, a mapping from the amplifier's distorted output back its undistorted input. Passing an undistorted input signal through this model generates the predistorted signal, which, when passed through the physical distortion source is transformed back to its original form—resulting in a nearly-linear output which closely matches the form of the input signal in spite of the heavy distortion introduced by the saturated amplifier. In general, predistortion offers a software-defined solution for managing the effects of amplifier distortion, providing increased flexibility and raising the prospect of on-site, active calibration.

From a systems perspective, this increased flexibility allows for more lax amplifier requirements as larger amplifiers (which must operate at a reduced power

level in order to stay within their linear region) may be substituted for smaller ones operating in saturation and aided by predistortion to linearize their output. For digital systems in particular, the predistortion method's potential to help reduce amplifier energy consumption requirements is especially relevant. DEE arrays must contend with energy distribution and thermal management at a system-level where each element is populated with a complete RF chain (digital transceivers, front-ends, high power amplifiers, etc.). Reducing the energy consumed by each element's amplifier helps to correspondingly reduce the weight of these system-level thermal concerns.

Critical to the success of digital predistortion is the designer's ability to accurately model an amplifier's distortion. Many methods exists for modeling amplifier behavior, but one method in particular (and variations thereof) sees extensive use for its relative simplicity and proven effectiveness. The Volterra series provides a generalized framework for modeling nonlinear systems with memory effects [9, 10, 17, 44, 53], and thus serves as an ample method for modeling amplifier distortion. Taking the following form:

$$y_v(n) = \sum_{k=1}^{K} v_k(n)$$

where: (4.1)

$$v_k(n) = \sum_{m_1}^{M-1} \cdots \sum_{m_k}^{M-1} \hat{\mathbf{h}}_k(m_1, \cdots, m_p) \prod_{l=1}^{k} x(n - m_l)$$

Where K defines the polynomial order (i.e. the number of Volterra kernels included in the model). Each Volterra kernel $v_k(n)$ then contains a unique set of coefficients $\hat{\mathbf{h}}_k(m_1, \cdots, m_p)$ which weigh their effects on the model's output.

The Volterra series is a class of functions defining linear mappings between sequences of input samples and equally sized output sequences. Each output

sample depends on weighted combinations of input samples—representing many unique states of the system as the input propagates through—enabling the model to reproduce higher-order effects generally ignored by more trivial methods. In general, increasing the polynomial order and/or the number of kernels for a given Volterra series will improve its capacity to accurately model phenomena exhibiting high-order effects, in turn widening its potential application space. However, raising these parameters corresponds to an exponential increase in the computational complexity and memory requirements for the model—a detraction of particular concern for applications calling for the model's deployment within embedded systems. Additionally, model training quickly becomes difficult as additional kernels necessitate the use of higher-dimensional optimization techniques. Finding the most effective balance between model complexity and accuracy has been a topic of significant interest as the benefits offered by digital predistortion have become indispensable.

These considerations led to the development of the memory polynomial model, a well-known simplification of the Volterra series consisting of a single kernel which operates on a signal's complex baseband samples in conjunction with their *exponentiated envelope* (i.e. complex baseband magnitude) [42], which takes the following form:

$$y(n) = \sum_{k=0}^{K-1} \sum_{m=0}^{M-1} \hat{\mathbf{h}}_{km} x(n-m) \left|x(n-m)\right|^k \quad . \tag{4.2}$$

The effectiveness of this method for modeling distortion caused by high-power RF amplifiers—inverse modeling thus enabling digital predistortion—is well established by the literature [17, 42]. Although Volterra models are known to be useful for modeling systems with both nonlinearities and memory effects, the memory polynomial model in particular has had previous success modeling phys-

ical amplifiers at complex baseband [15, 16, 43]. Additionally, the memory polynomial's relative simplicity lends itself to real-time application, as the memory and computational requirements for applying the model to streams of samples are sufficiently small.

Assuming well-formed training data, computing coefficients for the memory polynomial model is straightforward. Following a construct known in the literature as the 'delay matrix' [17], the permutations of the inner loop of Equation 4.9 (excluding $\hat{\mathbf{h}}_{km}$) may be formed into a two-dimensional structure, essentially providing a time-independent view of the memory polynomial's state for all cycles as a given input sequence passes through the model. This view makes way for the application of standard regression techniques for computing coefficients, as the problem is reduced to finding the best-fit linear mapping between the set of rows of the delay matrix and their pairs in the distorted training data.

Leveraging the delay matrix structure, the coefficients of the forward amplifier model are computed via a least squares relationship [8, 47, 61], where $\mathbf{y}_t$ is the distorted training vector and $\mathbf{X}_d$ is the delay matrix formed from the input vector $\mathbf{x}_t$. Therefore, following [17, 42] we have:

$$\hat{\mathbf{h}}_{forward} = \left(\mathbf{X}_d^H \mathbf{X}_d\right)^{-1} \mathbf{X}_d^H y_t \tag{4.3}$$

Assuming a well-formed solution, the resultant coefficient set $\hat{\mathbf{h}}_{forward}$ may be used with the model to simulate the distortion produced by the amplifier.

To generate a suitable approximation of the amplifier's behavior for a wide variety of inputs (or across some bandwidth), it is essential that this training waveform exhibits as many combinations of instantaneous power and frequency as possible. Such waveforms provide a rich dataset wherein many possible amplifiers states are exercised, generally improving the quality of models trained using such

data. It has been shown that Gaussian noise waveforms—filtered to an acceptable bandwidth—reliably accomplish this task [17].

Once established that the forward model provides a sufficient approximation of the amplifier's behavior the inverse model is calculated. Here, the training vector is formed into a delay matrix and used within the same least-squares relationship as the forward model, where $\mathbf{x}_t$ is the input waveform and $\mathbf{Y}_d$ is the delay matrix formed from the distorted training vector $\mathbf{y}_t$:

$$\hat{\mathbf{h}}_{inverse} = \left(\mathbf{Y}^H_{training}\mathbf{Y}_{training}\right)^{-1}\mathbf{Y}^H_{training}\mathbf{x}_{cal} \tag{4.4}$$

The two coefficient sets $\hat{\mathbf{h}}_{forward}$ and $\hat{\mathbf{h}}_{inverse}$ form models which act as inverse pairs—therein lying the crux of predistortion. By transforming a given waveform using the inverse model, the effect of the amplifier's distortion on the predistorted signal will only be to 'undo' the predistortion—resulting in the production of a waveform with reduced distortion at the HPA's output. Once adequate model coefficients have been derived, what remains is to apply the algorithm to base-band digital data prior to RF synthesis, yielding the predistorted signal which theoretically achieves the desired performance post-amplifier. In the HORUS system [46, 63], this operation will be performed in an FPGA in real-time (requiring custom logic), which will be discussed in the following section.

4.2 Real-Time Digital Predistortion in High Level Synthesis

For several of the next-generation phased array radar systems under active development at the University of Oklahoma's Advanced Radar Research Center (ARRC), the memory polynomial (MP) model has been selected as a foundation

for applying open-loop digital predistortion for the purpose of broadly optimizing array performance. In these systems, digital predistortion will be applied independently at each element, and will be computed in real-time within the FPGAs containing the systems' real-time digital back-ends.

In this section, we present our architecture for FPGA-based memory polynomial predistortion. Given the targeted systems' waveform diversity requirements, an ideal module would apply predistortion online, in real-time and with minimal buffering and latency. Meeting these requirements would thus ensure that the systems maintain a large degree of flexibility (e.g. supporting pulse-to-pulse waveform changes) while avoiding impacts to the system's maximum pulse-repetition-frequency (PRF) as well as other time-sensitive system-level constraints. To meet these requirements, a custom solution was developed using IntelFPGA's HLS toolchain to directly compute an MP model's effect on a given arbitrary length baseband signal in real-time, ultimately achieving optimal throughput with relatively low resource utilization and a high F_{Max} (i.e. maximum clock speed). In this section, details of this implementation will be described, and both simulation accuracy and post-fit resource usage will be evaluated.

4.2.1 Formulation of FPGA-based Real-Time Digital Predistortion

Field-programmable gate arrays (FPGAs) are a special class of computer chip which allow run-time reconfiguration of the connectivity and function of a broad set of interconnected programmable unit cells. These unit cells range in functional complexity from LUTs—which map a discrete number of inputs to a discrete number of outputs via a reconfigurable table, allowing for the implementation of arbitrary logic—to larger and more feature-rich unit cells such as DSPs (digital

signal processing units, or dedicated hardware units for computing arithmetic operations) and BRAMs—i.e. fixed-size dedicated block rams which can function as either FIFO (first in, first out) memory or as addressed random-access memory (RAM). Ultimately, these chips help reduce system development time and lend themselves to optimized design integrity (by enabling rapid simulation, testing, and debugging—including on deployed systems). Traditionally, firmware for FPGAs has been developed using hardware description languages (HDL) such as Verilog HDL, VHDL, and SystemVerilog—relatively low-level languages which compile into register-transfer level (RTL) logic ready for fitting, placement, and routing onto actual FPGA hardware.

More recently, FPGA manufacturers have focused greater attention on the development of *high-level synthesis* tools which see parallel logic described in terms of a high level modeling language in which operations are intuitively arranged by data-flow rather than in terms of logical compute structure (as typically required when writing HDL). When compiled, such designs are converted into either an intermediate hardware description language or directly into RTL. These tools often offer several advantages—namely, the design iteration timeline is greatly reduced as simulation and debugging may be accurately performed without requiring complex and oftentimes fraught HDL programming. Likewise, hardware optimizations such as pipelining, loop unrolling, etc. can either be inferred by the compiler or explicitly declared as part of the source file itself—enabling more optimized designs [2]. These tools do, however, suffer from platform-specific syntax and include variations in language semantics between FPGA suppliers and toolkit versions—all issues which can lead to poorly optimized and/or entirely broken results. With careful effort, however, these tools offer a convenient pathway to generating highly optimized FPGA components. One such environment

is IntelFPGA HLS, which leverages C++ as an HLS language. Since this work was developed for use on IntelFPGA (formerly Altera) Arria 10 FPGAs, the IntelFPGA HLS toolkit was targeted for adapting our predistortion algorithm to hardware.

A variety of philosophies and designs exist for hardware-based memory-polynomial predistortion [1]. In our application, the targeted systems' digital transmit chains generate two baseband samples in parallel at a 137.50 MHz sampling rate (F_s), supplying 275.00 MHz of total instantaneous bandwidth for RF synthesis. Applying predistortion at this datapath's internal sampling rate of 137.50 MHz would have thus required a polyphase architecture, which after investigation and considering our application's preference for direct computation could have resulted in a complex, resource-intensive, or unfavorably constrained design. However, the systems' relatively low F_s of 137.50 MHz allowed an alternate approach to be considered—that is, double-pumping, which doubles a component's throughput by clocking at twice the datapath's sampling rate, where the two samples are serialized and de-serialized prior to the component's input and after its output [27, 60].

Before this architecture could be realized, however, an FPGA system for applying predistortion must be developed capable of achieving an F_{Max} which meets or exceeds the required double-pumped frequency of 275.00 MHz, a design constraint which could be challenging to meet in practice. In general, F_{Max} refers to the highest estimated clock rate achievable by synchronous logic within a target FPGA, which is a function of the physical design placement and signal routing generated during design compilation [2, 41]. Ultimately, our architecture for memory-polynomial predistortion met the desired bandwidth with favorable resource usage and an F_{Max} which exceeds the required double-pumping clock

frequency. Likewise, our design achieved an application-favorable iteration interval (II) of 1—in general, this metric specifies the average number of clock cycles required to compute a single output sample given a single input sample [2, 57]. With an II of 1, the component incurs a constant latency due to its pipeline length but will otherwise continuously produce output samples every clock cycle without stalling.

Implementing this architecture in high-performance FPGA logic required substantial planning and optimization to achieve. Before this was possible, however, the compute structure of the memory polynomial was closely studied in abstract to identify potential areas for optimization. The model takes the following form, which sees an output sequence $y(n)$ computed in terms of the input sequence x and some coefficient matrix $\hat{\mathbf{H}}$:

$$y(n) = \sum_{k=0}^{\mathrm{K}-1} \sum_{m=0}^{\mathrm{M}-1} \hat{\mathbf{H}}_{km} x(n-m) \left|x(n-m)\right|^{k} \quad . \tag{4.5}$$

The model is controlled by two parameters: the polynomial order K, which determines the maximum order of exponentiated envelope considered by the model and intuitively relates to the level of non-linearity which the model may accurately estimate; and the memory order M, which determines how many previous input samples are considered by the model. Given the model's structure, the operation is inherently multi-dimensional as each output sample $y(n)$ depends on a full traversal of the memory polynomial's two summation operators to compute, suggesting complicated hardware implementation. However, breaking the algorithm into smaller pieces helped identify several reorganizations which—once applied—yielded an operation with a more straightforward mapping to hardware. Specifically, these changes helped to facilitate parallel computation while also making space for and encouraging pipelining.

To help build intuition regarding the model's compute structure, it was useful to consider the impulse response of the system, specifically tracking the model's internal state as sequences of samples pass through it. In this capacity, the model was first considered in terms of its component pieces. The inner kernel—which computes the input signal's "exponentiated envelope" [42]—was considered in isolation to simplify the apparent relationship between the model's coefficients and its internal state, that is

$$\begin{aligned} y\left(n\right) &= \sum_{k=0}^{\mathrm{K}-1} \sum_{m=0}^{\mathrm{M}-1} \hat{\mathbf{H}}[k,m] \cdot \hat{\mathbf{S}}_n[k,m] \\ &\text{where} \\ \hat{\mathbf{S}}_n\left[i,j\right] &= x\left(n-j\right)\left|x\left(n-j\right)\right|^i \quad . \end{aligned} \tag{4.6}$$

For a given sample number n, the 'state matrix' $\hat{\mathbf{S}}_n$ represents the memory polynomial's current 'state' including past inputs and their envelope estimates.

$$\begin{array}{c} \\ k{=}0 \\ k{=}1 \\ k{=}2 \end{array} \begin{array}{c} \text{State Matrix } \hat{\mathbf{S}} \\ \begin{array}{ccc} m{=}0 & m{=}1 & m{=}2 \\ \hat{\mathbf{S}}_{0,0} & \hat{\mathbf{S}}_{0,1} & \hat{\mathbf{S}}_{0,2} \\ \hat{\mathbf{S}}_{1,0} & \hat{\mathbf{S}}_{1,1} & \hat{\mathbf{S}}_{1,2} \\ \hat{\mathbf{S}}_{2,0} & \hat{\mathbf{S}}_{2,1} & \hat{\mathbf{S}}_{2,2} \end{array} \end{array} \longleftrightarrow \begin{array}{c} \text{Coefficient Matrix } \hat{\mathbf{H}} \\ \begin{array}{ccc} m{=}0 & m{=}1 & m{=}2 \\ \hat{\mathbf{H}}_{0,0} & \hat{\mathbf{H}}_{0,1} & \hat{\mathbf{H}}_{0,2} \\ \hat{\mathbf{H}}_{1,0} & \hat{\mathbf{H}}_{1,1} & \hat{\mathbf{H}}_{1,2} \\ \hat{\mathbf{H}}_{2,0} & \hat{\mathbf{H}}_{2,1} & \hat{\mathbf{H}}_{2,2} \end{array} \end{array} \tag{4.7}$$

As the state matrix and coefficient matrix share a common size, items in the state matrix directly pair with a single item in the coefficient matrix, see equation (4.7) for a hypothetical model with $\mathrm{K} = \mathrm{M} = 3$—as such, summing the elementwise multiplication of the coefficient matrix and a state matrix from a single time step yields a single output sample. An alternate way of understanding this operation is as a linear mapping between a point in $\mathrm{K} \times \mathrm{M}$ dimensional complex space (constructed from the flattened state matrix) and a single complex number, i.e.

$\mathbb{C}^{\mathrm{K}\times\mathrm{M}} \rightarrow \mathbb{C}^{1}$. In other words, the relationship between the model's two component matrices and the model's output is identical to the inner product between those two matrices, that is:

$$y\left(n\right) = \sum_{k=0}^{\mathrm{K}-1} \sum_{m=0}^{\mathrm{M}-1} \hat{\mathbf{H}}_{km} \hat{\mathbf{S}}_{km}$$

becomes

$$y\left(n\right) = \left\langle\ \hat{\mathbf{H}}, \hat{\mathbf{S}}_{n}\ \right\rangle\ , \tag{4.8}$$

where $\langle \hat{x},\ \hat{y} \rangle$ denotes the inner product of identically sized matrices $\hat{x}$ and $\hat{y}$. Critically, this inner product resolves to a single multiply-accumulate (MAC) operation, a structure which can be implemented efficiently in hardware as a highly-pipelined and highly-parallel multiply-accumulate tree.

With an appropriate, efficient structure selected for the most critical operation of model computation, what remained was to identify optimization points during state matrix assembly. To help clarify the assembly operation's dataflow, permutations of the state matrix were simulated (for some model with orders $\mathrm{K} = 3$ and $\mathrm{M} = 3$) as a test signal passes through. Consider the model supplied with this test signal—a scaled version of the causal unit impulse function:

$$x = \sqrt{2} \cdot \delta[n] = \begin{cases} \sqrt{2}, & \text{if } n > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.9}$$

The unit impulse was selected as a test sequence both for simplicity and to ensure that adjacent input samples are decoupled from one another. Likewise, scaling the impulse by $\sqrt{2}$ ensured that the exponentiated envelope term was unique for all values of k. In equation (4.10), the state matrices for each model iteration $n \in \{-1, 0, \cdots, 3\}$ are shown, where $n = -1$ represents the model's initial state:

$$\hat{\mathbf{S}}_n : \begin{matrix} \text{n=-1} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{matrix} \rightarrow \cdots \qquad (4.10)$$

$$\cdots \begin{matrix} \text{n=0} \\ \begin{bmatrix} 2 & 0 & 0 \\ 4 & 0 & 0 \\ 8 & 0 & 0 \end{bmatrix} \end{matrix} \rightarrow \begin{matrix} \text{n=1} \\ \begin{bmatrix} 0 & 2 & 0 \\ 0 & 4 & 0 \\ 0 & 8 & 0 \end{bmatrix} \end{matrix} \rightarrow \begin{matrix} \text{n=2} \\ \begin{bmatrix} 0 & 0 & 2 \\ 0 & 0 & 4 \\ 0 & 0 & 8 \end{bmatrix} \end{matrix} \rightarrow \begin{matrix} \text{n=3} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

Consider only the top row of each state matrix in equation (4.10), representing the $k = 0$ kernel of the memory polynomial over time. At model iteration $n = 0$, the first sample of the scaled unit impulse is assigned to the state matrix as $\hat{\mathbf{S}}_n[0, 0] = x[0] = \sqrt{2}$. Moving on to the $k = 1$ and $k = 2$ kernels, we find that the same logic is present except that powers of the envelope term are supplied, that is $\hat{\mathbf{S}}_n[k, 0] = x[0]^k = \sqrt{2}^k$. In subsequent iterations, the columns of the state matrix shifted from lower index positions to higher index positions. In other words, each row of the state matrix acts as a delay line.

When paired with the inner product, it is clear that each polynomial kernel performs an operation identical to a M-length finite impulse response (FIR) filter—that is, a delay line with coefficient multipliers for each delayed sample which are accumulated to produce a single output sample. These FIR filters act in parallel, operating on increasing powers of the signal envelope before being combined via summation. This equivalency stands as striking evidence that—if organized correctly—an efficient architecture for directly computing the memory polynomial is possible.

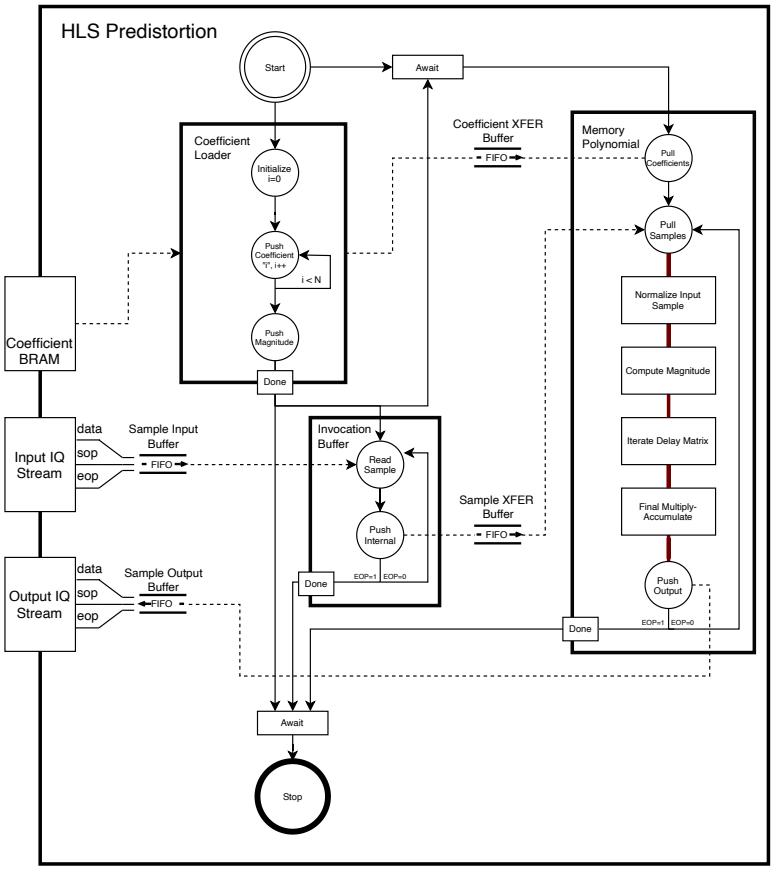


Figure 4.2: Block diagram showing the architecture of the memory polynomial HLS component.

4.2.2 Implementation in IntelFPGA High Level Synthesis

Leveraging insights from Section 4.3, a version of the memory polynomial model was implemented for an IntelFPGA Arria 10 (A10) FPGA target. For reduced syntactical burden and faster simulations, IntelFPGA's high-level synthesis tools—which adapt C++ into a viable simulation and synthesis language—were used. Our design can accommodate memory polynomial models with maximum parameters $K = 4$ and $M = 4$, which were found to be optimal (at the intersection between resource usage and predistortion performance). Once the design's numerical accuracy was validated, two distinct optimization strategies were applied to achieve ideal throughput and reduce resource usage—compute structure opti-

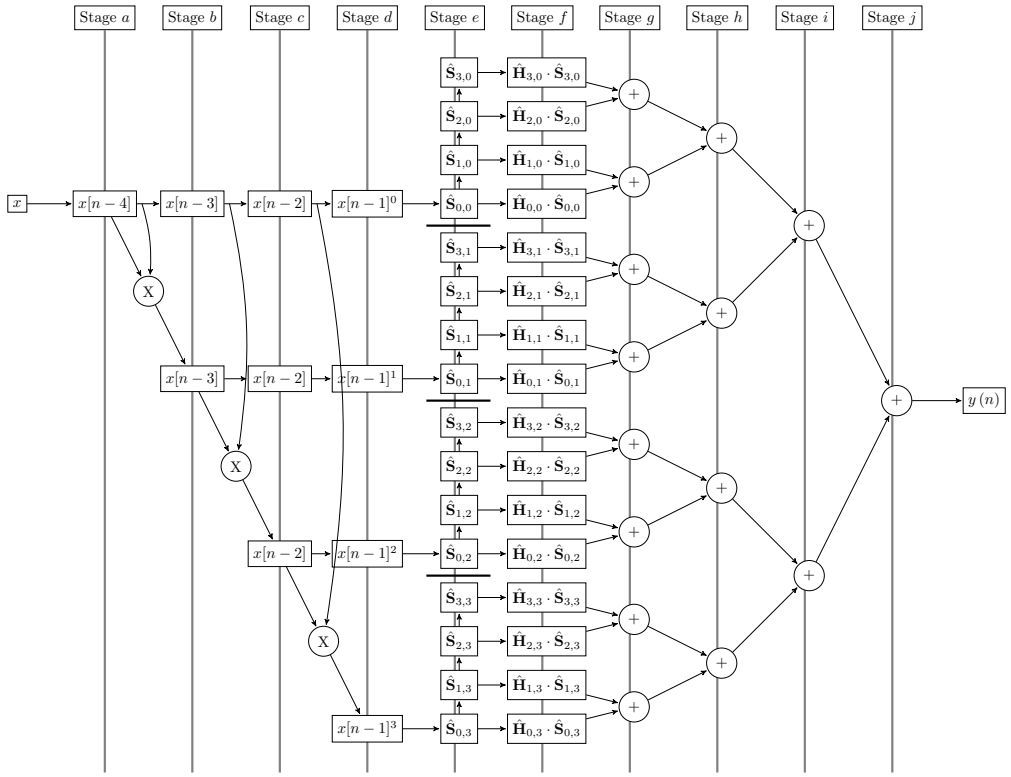


Figure 4.3: Dataflow and pipeline diagram for K = 4 and M = 4 [24]. Reprinted from Herndon and Yeary (2022) © 2022 IEEE

mization, which primarily aimed to parallelize and pipeline the critical compute path, and architectural optimization which aimed to adjust the design in broad strokes for favorable improvements.

Architectural Optimizations

Originally, the memory-mapped coefficient storage was implemented in block RAM which then connected directly to the multipliers. This resulted in a large and unsustainable fan-out when feeding the coefficients to the final MAC which caused the design to fail timing. In this instance, forcing HLS to implement memory-mapped coefficient storage in distributed registers did not resolve this issue. Although the specific dynamic which caused this failure was not well

understood (as it appeared to be multifaceted), one theory was that the specific HLS optimizations which were performed to generate the addressable memory were incompatible with the large fan-out necessary to make the design work. Regardless, to sidestep this issue, the memory mapped coefficient storage block ram was divorced from the coefficient fan-out. Now, distributed registers drive the MAC which are supplied with coefficients via an asynchronous pipe during an initialization stage each time the component is invoked. This optimization relaxed constraints on the coefficient storage block RAM (which can now accommodate multi-cycle reads and writes) and ultimately resolved the component's timing problems. Likewise, it improved the design's scalability, as the block RAM may now be of arbitrary size without interfering with the compute operation's internal timing.

Compute Optimizations

The critical compute path (*Memory Polynomial* in Fig. 4.2) relies on fixed point arithmetic with various application-specific precisions to implement equation (4.8). One important compute optimization thus centered on optimizing numerical precision against resource usage. The design was parameterized such that each distinct numerical role (state matrix, delay line, coefficient, etc.) relied on an independent fixed point type declaration with potentially unique precisions and—complexity which is simple to manage within IntelFPGA'S HLS toolchain. The tradespace between these parameters and the model's overall numerical accuracy was optimized by recompiling with various parameter sets, comparing their simulated outputs against a floating point ground truth model generated in Python, and selecting the parameter set with the least precision which still achieved favorable error metrics.

To encourage the high-level synthesis compiler to generate favorable logical structures in the memory polynomial component's critical compute path, the operation was divided into partial products wherever possible. Applying this reorganization scheme allowed each partial operation to be parallelized and optimized separately, then later chained together favorably to encourage pipelining. To further encourage optimal structures, several of the operation's partial products were implemented leveraging unconventional C++ loop structures and IntelHLS-specific preprocessor directives. Fig. 4.3 diagrams an approximation of the achieved compute-path pipeline[1], demonstrating the relationship between its partial products and visualizing motivating principles from Section 4.3. For the remainder of this section, several of these optimal structures will be discussed along their associated code snippet. Many of these structures rely on the HLS preprocessor directive `#pragma unroll` which forces full loop unrolling, generating parallel logic for each loop iteration. For this operation to be successful, there must not be any data dependencies between iterations.

Envelope Calculation

To sidestep the data dependency inherent in power-of-envelope term generation and reduce resource consumption, higher order enveloped terms are computed via a pipelined approach rather than direct parallel computation. In this stage of the critical compute path, the first order magnitude term is computed and shifted into delay matching logic for later use.

[1]Many computational details have been excluded for brevity.

```
// Delay match x with multi-clock latency magnitude samples
delay_readin[0] = x;

// abs(x) ^ 2
delay_mag[0] = x.real()*x.real() + x.imag()*x.imag();
```

Code 4.1: Envelope Calculation

Delay Lines, Matching, and Pipeline Assist Logic

Sample reorganization and delay matching for the compute path is managed by the following two loops. In particular, the code below represents the first loop (which includes the logic for generating the delay line bank in stage e) and in part implements a one-dimensional delay line with boundaries every M samples corresponding to a flattened version of the 2D shift shown in equation (4.10).

```
// Shift delay lines
int i;
#pragma unroll
for(int m=MP_MAX_M-1; m>=0; --m){
#pragma unroll
    for(int k=MP_MAX_K-1; k>0; --k){
        // Do not shift across region boundary.
        i = m*MP_MAX_K + k;
        delay_matrix[i] = delay_matrix[i-1];
        // magnitude delay line
        delay_kmag[i] = delay_kmag[i-1];
    }
}
```

Code 4.2: Delay Lines

Likewise, delay-matching logic is generated by the following which corresponds partially to stages a–d

```
// Magnitude assembly & delay matching
#pragma unroll
for(int i=MP_MAX_K-1; i>0; --i){
    // delay complex samples
    delay_readin[i] = delay_readin[i-1];

    // delay raw magnitude samples
    delay_mag[i] = delay_mag[i-1] * 1;

    // assemble powers of each magnitude leveraging raw magnitude delay line
    delay_kmag[i*MP_MAX_K] = delay_kmag[(i-1)*MP_MAX_K] * delay_mag[i];
}
```

Code 4.3: Delay Matching and Pipeline-Assist Logic

State Matrix Generation

As detailed in equation (4.10), each new sample shifted into the component corresponds to a new column shifted into the state matrix, where each row represents a distinct power-of-envelope term—the following HLS snippet generates this logic,

corresponding to stages *d*–*e* in Fig. 4.3. Note: although state matrix samples are generated here, they are delayed elsewhere—specifically, by the logic detailed in Section 4.2.2.

```
// Iterate state matrix
#pragma unroll
for(int i=0; i<MP_MAX_K; ++i){
    // Multiply raw sample delayed by constant value "MP_MAX_K"
    // by that sample's magnitude to the kth power, selected
    // from the delay_kmag structure. Each "k" applies a one
    // clock cycle delay, which is managed by forcing the
    // constant delay "MP_MAX_K" and selecting values from
    // delay_kmag such that all samples are time-aligned.
    delay_matrix[i*MP_MAX_K] = delay_readin[MP_MAX_K-1] * delay_kmag[i*MP_MAX_K + MP_MAX_K-i-1];
}
```

Code 4.4: Stages *d*–*e*: compute new state matrix samples

Inner Product

Lastly, the final operation—which computes the inner product between the state and coefficient matrices—was implemented as a multiply-accumulate (MAC) operation. This maps efficiently to hardware as a multiply-accumulate tree, which is inferred by the HLS compiler from the following structure representing stages *f*–*j* of Fig. 4.3

```
// Apply coefficients to delay matrix, sum to form output sample
y_pd = 0;
#pragma unroll
for (int i=0; i<MP_MAX_K*MP_MAX_M; ++i){
    y_pd += delay_matrix[i] * coeffs[i];
}
```

Code 4.5: Stages *f*–*j*: optimized MAC tree for computing inner product.

4.2.3 Simulation and Results in-Hardware

After optimization, our HLS implementation of the memory polynomial model achieved favorable results in simulation (simulation error analysis shown in Fig. 4.5). With substantial pipelining, the component achieved an optimal II of 1. Although Fig. 4.3 demonstrates the generic pipeline model targeted during the design process, in actuality the synthesized logic includes many more pipeline stages. In total, the internal pipeline has a total latency of 123 clock cycles, sug-

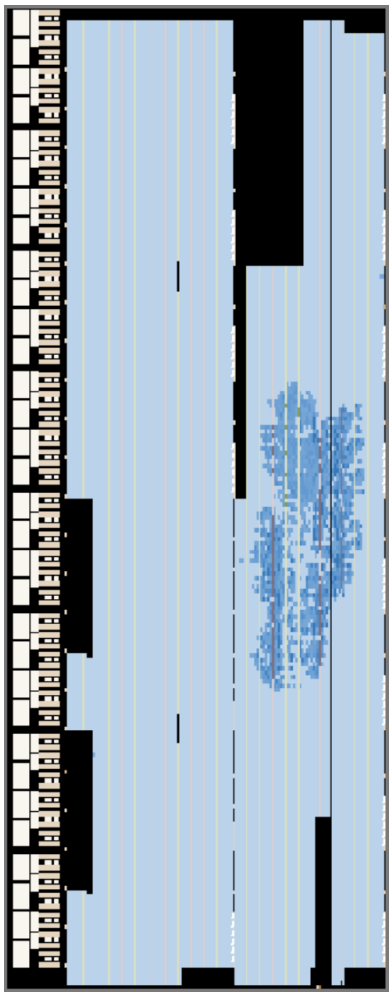


Figure 4.4: Post-fit floorplan for the component (including supporting testbench logic) when compiled for the A10 [24]. Reprinted from Herndon and Yeary (2022) © 2022 IEEE

gesting an equivalent number of pipeline stages when considering the achieved II. Likewise, to meet the requirements of the module's target parent chip (IntelFPGA Arria 10, model 10AX057N2F40E2SG), the component's desired maximum frequency of 275.00 MHz was successfully achieved, allowing the component to be double-pumped to meet polyphase baseband throughput requirements. Table 4.1 shows the post-compilation resource usage of our component, and Fig. 4.4 shows the FPGA's post-fit floorplan. In the Arria 10, the resources available within the

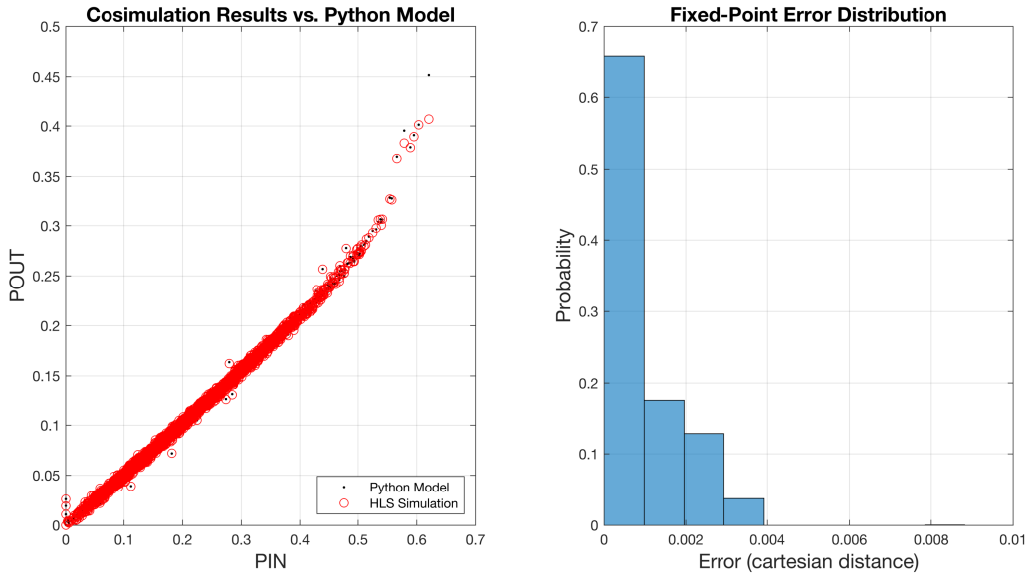


Figure 4.5: Comparison between the HLS component's co-simulation test results and the Python model simulation [24]. Reprinted from Herndon and Yeary (2022) © 2022 IEEE

FPGA are divided between ALUTs, FFs, RAMs, DSPs, and MLABs, where:

- ALUTs are look-up table units,
- FFs are flip-flop units,
- RAMs are dedicated block ram units,
- DSPs are digital signal processing units, and
- MLABs are distributed memory units.

	ALUTs	FFs	RAMs	DSPs	MLABs
# Units	16480	23156	3	106	192
% Total	3.49 %	2.38 %	1.35 %	0.11 %	6.98 %

Table 4.1: Post-Compilation Resource Usage

In total, the component's relatively low resource consumption was sufficient for

our purposes, and enabled four parallel components to be instantiated corresponding to the four independent baseband channels managed by the A10 unit.

4.2.4 Future Work

In its current state, the HLS component is acceptable in both form and function for our target applications—however, reducing resource consumption and one potential new feature have been targeted for future efforts (which will focused primarily on optimization). Although the component's resource usage is acceptable in its current state (shown in Table 4.1), ALUT and FF usage could likely be reduced. After limited examination, much of the excess in these two unit types appears tied to the asynchronous coefficient loading process described in Section 4.2.2, a design choice which was required to meet timing. Although this timing optimization resulted in a disproportionate increase in resource usage, this negative trade-off appears disconnected from the compute path and could likely be corrected via close inspection of the component's usage statistics and slight modifications to the coefficient bus architecture. On a more positive note, this timing optimization created an opportunity for a novel feature should it ever be desirable. As the meta-function of coefficient loading is now divorced from the DPD compute operation and since the component leverages the component invocation interface (an asynchronous control scheme offered within IntelFPGA HLS [2]), the component was developed to operate on each new pulse in complete isolation with no persistent state. Each component invocation loads coefficients from a multi-cycle latency memory-mapped RAM block which is (as of yet) sized to fit a single model configuration. With limited design work, the component's coefficient storage could be resized to accommodate more than one DPD model and an index term could be added to the component's invocation interface with minimal

impact to resource usage or logical complexity. This would allow for pulse-to-pulse distortion model changes based on external variables (e.g. temperature), a potentially valuable capability which could enable performance improvements across an even wider range of environmental conditions.

4.2.5 Summary

This section details the architecture of an FPGA-based high-level synthesis component for applying the memory polynomial model to data streams in real-time. By analyzing the model's compute structure and data flow, several critical optimization points were found and exploited, ultimately resulting in a design which achieved optimal throughput (II=1) while retaining numerical accuracy, favorable model performance, and acceptable resource consumption.

In addition to novel compute systems for applying predistortion, next-generation systems will correspondingly require new training methods for performing in-situ calibration. In the following section, a series of experiments are detailed which sought to explore the feasibility of training predistortion models using baseband IQ data captured via mutually coupled receiver elements adjacent to a saturating transmit element.

4.3 Digital Predistortion Model Training via Mutual Coupling: A Case Study

For systems to support on-site recalibration of predistortion models, there must be feedback paths in place for capturing model training data. Several architectures currently exist for enabling this behavior. One option uses high-power switches after each amplifier's output to create loopback paths during calibration.

Another option sees RF couplers at each amplifier's output performing much the same function as the switched option, though passively. Although effective, such options increase the complexity of front-end modules and exclude additional distortion effects which may be generated after the feedback point from the model training process. In this section, mutual coupling between active array elements is discussed as another potential option for creating feedback paths for measuring predistortion training data—one which potentially avoids the drawbacks inherent to hardware solutions while offering its own advantages.

In an array environment, mutual coupling refers to the set of all inter-element interactions wherein energy radiated from one element is collected and re-radiated by adjacent elements and vice versa. Mutual coupling has been shown to be an important phenomenon that allows for calibration, for instance [5, 6, 55] and others. In this section, a new potential application for this phenomenon is discussed, i.e. as a mechanism for providing feedback for measuring transmitter front-end distortion for elements within a DEE array. By transmitting a saturated signal from a single element and capturing couplings from several adjacent receiver elements, it is thought that the distortion affecting the transmitting element may be characterized using those data (as each receiver observes the same distorted radiative source) in spite of differences between receiver elements and the imperfect nature of the measurement. This method opens the possibility of modeling each element's complete RF chain from DAC to antenna, theoretically allowing the memory polynomial model to account for additional sources of distortion in the system that would otherwise be missed.

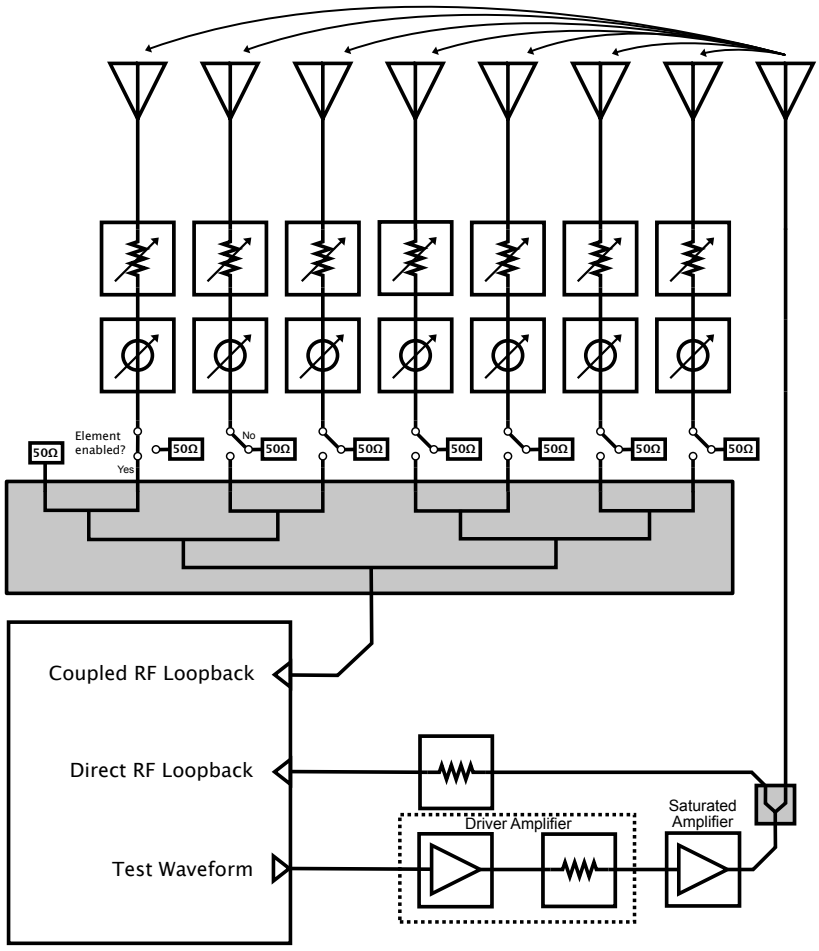


Figure 4.6: Diagram of the research platform, showing the relationship between the observation and coupled loopback channels [25]. Reprinted from Herndon, Yeary, and Palmer (2020) © 2020 IEEE

4.3.1 Experimental Procedure and Results

As part of the process of working towards a practical implementation of digital predistortion for use in the next-generation phased array radars in development at the ARRC, a research platform has been developed for investigating possible methods for training and testing memory polynomial predistortion. Rather than demonstrating the effectiveness of these methods in reducing spectral regrowth produced in an isolated high-power amplifier [17], this platform aims to emulate

a practical system with the understanding that future DEE systems will require amplifier characterization from imperfect measurements.

The research platform is a simple phased array (diagram seen in Figure 4.6) constructed from a collection of fundamental radar components (TR Module, HPA, etc.) in order to emulate a functional system. Control software was developed using RadarControllerAPI to allow easy control via RadarControllerGUI (see Chapter 2), which helped automate experiments performed with the system, and included hooks for applying DPD models to transmit waveforms as well as ScanModes for visualizing amplifier linearity in real time. The forth row within an 8×8 array developed within the ARRC[2] acts as an 8×1 linear array, with all other array elements terminated with 50 Ω loads. This test array is mounted facing straight up to minimize environmental reflections during radiative testing. A single element from this set is selected as the transmitter element by manually connecting it to the amplifier's output. The seven remaining elements were then attached to a digitally-controllable phase-amplitude controller (PAC) board which multiplexes their collected energy to a single port—allowing their couplings with the transmit element to be measured in isolation from the other receiving elements. To emulate the digital backend of a single element within a digital array, a TR module developed within the ARRC leveraging an Analog Devices AD9361 digital transceiver chip is used to synthesize and measure arbitrary S-band RF. To generate a sufficiently distorted test signal, the transmitter port of this device was connected to a series of two high-power amplifiers—the first acting as a driver amplifier, followed by a second amplifier driven into saturation by the first. Next, a power divider splits the amplifier's output between an observation loopback channel and a single radiating element on the linear test array. The selected transmit element was changed over the course of the experiment to

[2]For further discussion of a similar 8×8 array, see [14]

capture measurements from different locations on the array operating subject to differing array geometries. The loopback channel acts as the experiment's ground truth by providing a direct observation of the distorted amplifier's output, useful for judging the accuracy of predistortion models generated using the coupled measurements. Later, two experiments performed using this platform will be discussed—one where the array was tested outside, and the other where it was tested within the ARRC's farfield RF anechoic chamber. Both yielded interesting insights into the nature of the problem.

As maximum transmit pulse width (τ) limitations exist in practical systems (effectively limiting the maximum length of any one training waveform), there is strong motivation to explore ways of combining sets of trials into a single, rich dataset for use training the predistortion models. The TR module used in this project specifically has a maximum transmit length of $\tau = 100\ \mu s$ with a baseband sampling frequency of $F_s = 30$ MHz, facts which together limit the maximum training dataset length to 3000 samples. It was discovered early in the research process that datasets of this size were insufficient to accurately characterize the amplifier, and resulted in unusable predistortion models. Qualitatively, the point density in the $P_{pin} \rightarrow P_{out}$ functional space from these short datasets is sparse compared to results generated using longer training waveforms as in [17], a result of the limited available random samples spreading out too broadly within the output space. Using longer training waveforms in effect 'fills in' more of these points, generating a more complete picture of the amplifier's characteristic curve and providing an improved target for model fitting. With the aim of reproducing the results seen in the literature which drew from longer datasets [17], ten different waveforms were generated using complex Gaussian noise then filtered to 10 MHz of bandwidth. To emulate larger contiguous datasets, these trials were captured

separately then stitched together in post-processing. Using four different transmit element positions, the couplings from the transmit element to each of the seven receiver elements were captured using this method, generating a breadth of data for use in model training.

Leveraging these data and following methodologies detailed in [17], predistortion models were generated by directly implementing the formulas outlined in Section 4.3 in a MATLAB script. One significant aspect of model training not yet addressed, though, is the process for selecting the most effective parameters for the memory polynomial model (polynomial and memory orders K and M for both the forward and reverse models). In lieu of a more deliberate approach, a brute-force algorithm was developed to permute through a predefined coefficient space. For each combination of parameters, the input signal $\mathbf{x}_t$ was passed through the forward model $\hat{\mathbf{h}}_{forward}$ and scored against the training signal $\mathbf{y}_t$ by finding the mean-squared error between the two signals. Similarly, $\mathbf{y}_t$ was passed through the predistortion model $\hat{\mathbf{h}}_{inverse}$ and scored against $\mathbf{x}_t$ to determine how effectively the trained predistortion model could undo distortion effects observed in $\mathbf{y}_t$. From these trials, the most effective coefficients were those which minimized a combination of the scores generated from the forward and inverse models.

Early in development, this process for generating and verifying predistortion models was tested using data captured from the direct observation loopback channel. The results of this trial (seen in Figure 4.7) show reduced spectral spreading and improved $P_{pin} \rightarrow P_{out}$ linearity, and in doing so demonstrate the method's effectiveness. Following these initial efforts, two experiments were performed to generate training data from mutual coupling measurements using the previously described procedure.

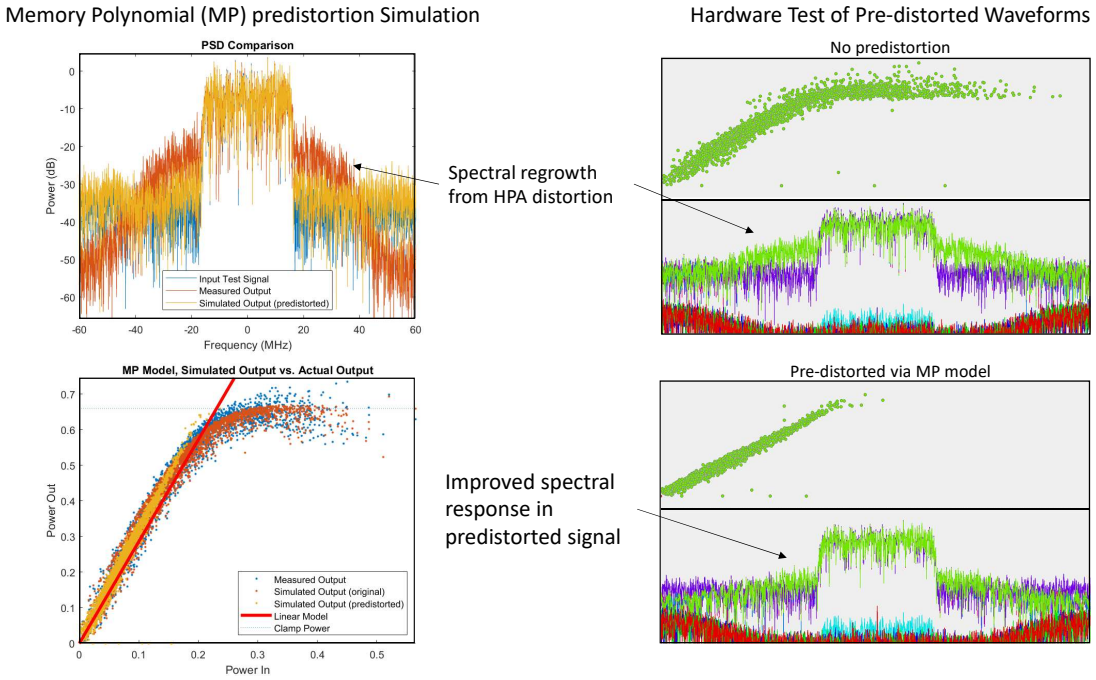


Figure 4.7: Successful test of predistortion using a model trained from data measured via the ground truth direct observation channel [25]. Reprinted from Herndon, Yeary, and Palmer (2020) © 2020 IEEE

Experiment 1

In the first experiment, the test platform was placed outside away from obstacles. Figure 4.8 shows a series of ten trials captured from the test platform, where a single element radiates and was observed from only one receive element. The reference loopback channel has the expected shape of a characteristically distorted, highly-saturated amplifier. The coupled loopback, promisingly, has the same general shape as the observation channel but appears corrupted by additional distortion from an unknown source. In experiment 2, a second trial was performed in an attempt to isolate these unknown distortion effects.

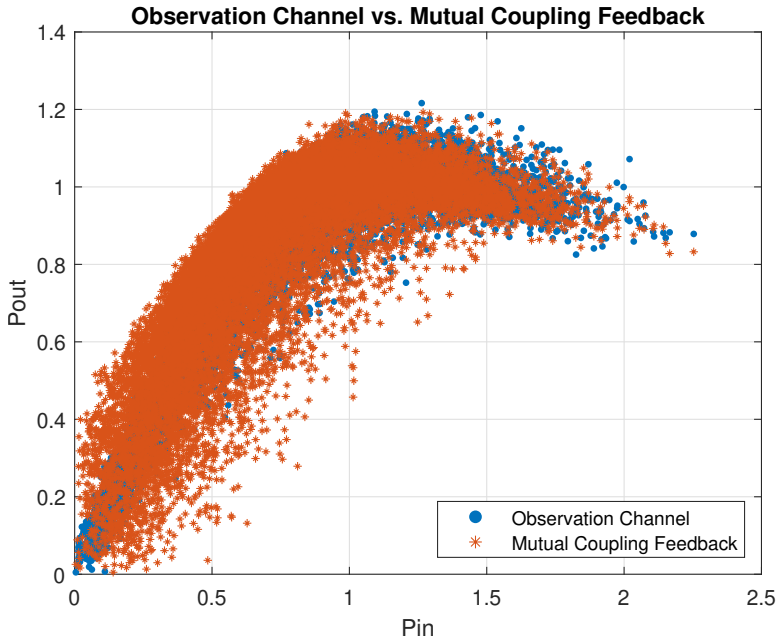
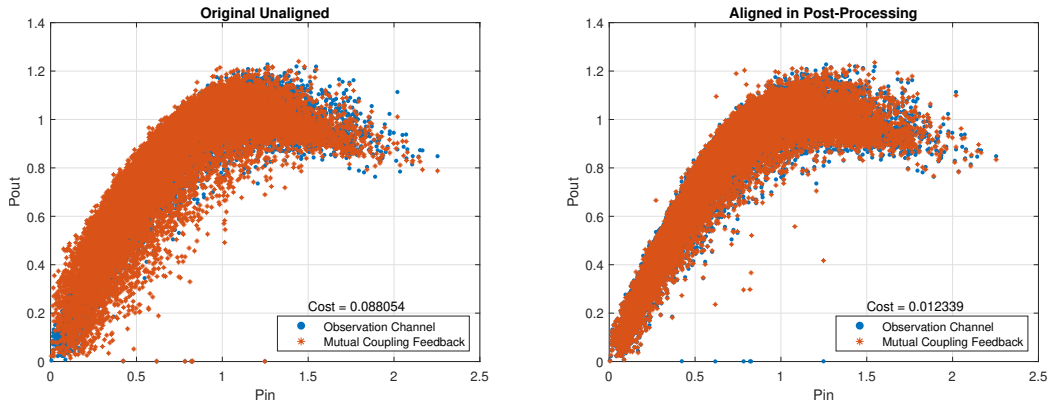


Figure 4.8: Using data from experiment 1, power curves observed from the direct loopback channel compared to those from the mutual coupling feedback channel [25]. Reprinted from Herndon, Yeary, and Palmer (2020) © 2020 IEEE

Experiment 2

In an effort to account for the significant difference in shape between the observation and coupled channels from Section 4.3.1, a second experiment was performed with the research platform placed in the ARRC's farfield chamber to help attenuate environmental effects, narrowing the field of causes which could explain the discrepancy. Although marginally improved, the structure of the data captured in this experiment (seen in Figure 4.9a) remains highly similar to the data captured during the first experiment, suggesting environmental effects were not the most significant source of distortion. Further investigation of the data found that the distortion observed in both experiments was primarily a result of time misalignment between P_{pin} and P_{out} leading to the reference and training waveforms de-correlating.

To find and correct for this unknown time offset, a cost function was defined as



(a) Before time alignment.

(b) After time alignment.

Figure 4.9: Using data from experiment 2, a comparison between raw and aligned $P_{pin} \rightarrow P_{out}$, showing improved correlation and substantially reduced cost between the observation and coupled channels after alignment [25]. Reprinted from Herndon, Yeary, and Palmer (2020) © 2020 IEEE

the average Cartesian distance between 2D points in the observation and coupled channels' functional spaces. This metric was then minimized across a range of artificially time-shifted versions of the coupled data to find the time offset which best aligns the two datasets. Shown in Figure 4.9b with their corresponding costs, the cost-minimized coupling data shows a marked improvement in its apparent correlation with the observation loopback channel (e.g. point pairs match more closely between the two datasets) as compared to the unaligned data, an observation reinforced by the calculated costs which saw close to an 85% decrease after alignment.

4.3.2 Summary

This section specifically addressed efforts to explore methods of supporting array calibration by combining aspects of mutual coupling and traditional non-linear digital predistortion. Rather than relying on switchable RF paths to provide training signals for predistortion algorithms as many radars do, mutual coupling

signals are captured from elements adjacent to the transmit element under test, providing one or more feedback paths for capturing datasets for training predistortion models for each element. Using the research platform described therein, measurements of a highly saturated amplifier's output were observed simultaneously from both mutual coupling between active array elements and from a loopback channel, allowing for direct comparisons to be drawn between those datasets. This platform and the methodologies described therein aim to provide a foundation for future studies on this topic.

4.4 Closing Remarks

Next generation digital phased array radars will rely on a variety of optimization methods to achieve performance targets. Digital predistortion is one such calibration method, which in the case of DEE arrays will ensure that each element is able to achieve near-linear behavior in saturation, optimizing the maximum power-on-target the array is able to generate without exceeding waveform distortion tolerances.

Chapter 5

Conclusion

The next generation of phased array radars will foster new paradigms of array functionality, with digital-at-every-element arrays in particular offering many advantages over more conventional array architectures. However, the promise offered by these systems requires overcoming proportionally complex engineering challenges before their vision can be realized.

In Chapter 2, an architecture for generalized high-level software for managing a set of networked controller nodes was described. At its core, this software package provides a basis for controlling sets of software daemons distributed locally or across a network, enabling cohesive and repeatable control of both single or multi-nodal systems. Originally developed for the CPPAR project to allow control of a bistatic system (see Chapter 3), additional efforts were taken to divorce platform specific logic from the original codebase, resulting in a capable software package for conducting research with experimental weather radar systems which has seen use in several other projects.

In Chapter 3, the calibration procedure for the Cylindrical Polarimetric Phased Array Radar (CPPAR) system was discussed in detail. This system makes use of a remote farfield calibration horn in combination with synchronous GPS triggering to capture bistatic measurements with precisely aligned pulse envelopes, but

decidedly non-coherent phase. Environmental differences between the two nodes' reference clock generators resulted in a time-varying frequency offset between the two transceivers' local oscillators, which introduced an arbitrary phase drift observable in the baseband IQ data, thereby concealing relative phase differences between elements in the measured patterns. To manage this effect, the system's mechanical pedestal, rapid beamformer weight switching, and precise element-level controls were leveraged to design a unique interleaved measurement scheme which exploits the array's cylindrical geometry to generate patterns for all elements within a single scan. This method was designed to ensure that a subset of elements were always observable from the farfield node, thus providing a phase-tracking target which could be measured in post-processing. By quantifying the precise sampling geometry for the CPPAR's array for the duration of the measurement rotation, we were able to achieve accurate post-processing spatial alignment of the measured patterns. Likewise, through two strategic reinterpretations of the two-dimensional pattern matrices generated by this measurement, a so-call *motion-invariant* view of the bistatic inverse pattern was formed, providing a basis for estimating bistatic phase drift while attenuating phase changes induced by the array's physical rotation. Finally, the bistatic phase drift estimate generated by this process was subtracted from the spatially aligned element patterns to yield sufficiently accurate power and phase element patterns for the array's 48 horizontally and 48 vertically-polarized elements, enabling calibration.

Continuing the theme of calibration, Chapter 4 details theory and system-level implications for digital predistortion (DPD) as applied to phased array radar systems. In this chapter, the memory polynomial technique for modeling 'nonlinear systems exhibiting memory effects' is detailed. Later, we detail our implementation of the memory polynomial within the IntelFPGA High-Level

Synthesis toolkit. By analyzing the model in abstract and observing aspects of its dataflow, a practical pipelined architecture was defined capable of computing the model's output in real-time. Likewise, we discussed specifics of how this component was integrated into our system and detailed our rational for a variety of design decisions, including the use of double-pumping to achieve our desired throughput given our application's two-sample polyphase transmit datapath. Lastly, we addressed our progress conducting a research project which aimed to explore methods for combining aspects of mutual coupling and traditional non-linear digital predistortion. Rather than relying on RF switches or RF couplers within a system's RF front-end to observe nonlinear distortion introduced post-amplifier, this section details an alternate design which sees the novel application of mutual coupling measurements for the purpose of training predistortion algorithms. Using the research platform described therein, measurements of a highly saturated amplifier's output were observed simultaneously from both mutual coupling between active array elements and from a loopback channel, allowing for direct comparisons to be drawn between those datasets. This platform and the methodologies described therein aimed to provide a foundation for future studies on this topic.

Taken together, this thesis details a variety of practical engineering challenges related to phased array system design, with particular emphasis placed on methods for control, calibration, and performance optimization of these systems. Therefore, the efficacy of phased array radar research gains momentum.

Bibliography

[1] Designing polyphase DPD solutions with 28-nm FPGAs. Technical report, Altera Corporation, San Jose, CA, USA, 2012.

[2] Intel® high level synthesis compiler pro edition reference manual. Technical report, Intel Corporation, Santa Clara, CA, USA, 2020.

[3] C. Andrich, A. Ihlow, W. Kotterman, N. Beuster, and G. Del Galdo. Using software defined radios for baseband phase measurement and frequency standard calibration. In *2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–5, May 2017.

[4] N. C. Athanasopoulos, N. K. Uzunoglu, and J. D. Kanellopoulos. Development of a 10 GHz phased array cylindrical antenna system in corporating if phase processing. *Progress In Electromagnetics Research*, 59:17–38, 2006.

[5] H. M. Aumann, A. J. Fenn, and F. G. Willwerth. Phased array antenna calibration and pattern prediction using mutual coupling measurements. volume 37, pages 844–850. IEEE, 1989.

[6] D. Bekers, R. van Dijk, and F. van Vliet. Mutual-coupling based phased-array calibration: A robust and versatile approach. In *2013 IEEE International Symposium on Phased Array Systems and Technology*, pages 630–637. IEEE, 2013.

[7] J. K. Bekkeng. Calibration of a novel MEMS inertial reference unit. *IEEE Transactions on Instrumentation and Measurement*, 58(6):1967–1974, June 2009.

[8] A. Ben-Israel and T. N. Greville. *Generalized Inverses: Theory and Applications*, volume 15. Springer Science & Business Media, 2003.

[9] S. Benedetto, E. Biglieri, and R. Daffara. Modeling and performance evaluation of nonlinear satellite links-a Volterra series approach. Number 4, pages 494–507. IEEE, 1979.

[10] S. Boyd and L. Chua. Fading memory and the problem of approximating nonlinear operators with Volterra series. volume 32, pages 1150–1161. IEEE, 1985.

[11] G. Bucci and C. Landi. Measurement techniques for the characterization of wireless communication systems in time domain. In *IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics (Cat. No.01CH 37188)*, volume 2, pages 1265–1270 vol.2, May 2001.

[12] C. Chang, M. Jin, and J. Curlander. Squint mode SAR processing algorithms. In *Proc. IGARSS*, volume 89, pages 1702–1706, 1989.

[13] M. I. Dessouky, H. A. Sharshar, and Y. A. Albagory. Efficient sidelobe reduction technique for small-sized concentric circular arrays. *Progress In Electromagnetics Research*, 65:187–200, 2006.

[14] J. D. Díaz, J. L. Salazar, J. A. Ortiz, C. Fulton, N. Aboserwal, R. Kelley, and R. Palmer. A dual-polarized cross-stacked patch antenna with wide-angle and low cross-polarization for fully digital multifunction phased array radars. In *2016 IEEE International Symposium on Phased Array Systems and Technology (PAST)*, pages 1–4. IEEE, 2016.

[15] L. Ding, G. T. Zhou, D. R. Morgan, Z. Ma, J. S. Kenney, J. Kim, and C. R. Giardina. A robust digital baseband predistorter constructed using memory polynomials. volume 52, pages 159–165. IEEE, 2004.

[16] Z. Dunn, M. Yeary, C. Fulton, and N. Goodman. Memory polynomial model for digital predistortion of broadband solid-state radar amplifiers. In *2015 IEEE Radar Conference (RadarCon)*, pages 1482–1486. IEEE, 2015.

[17] Z. Dunn, M. Yeary, C. Fulton, and N. Goodman. Wideband digital predistortion of solid-state radar amplifiers. volume 52, pages 2452–2466, October 2016.

[18] B. Friedlander. The MVDR beamformer for circular arrays. In *Conference Record of the Thirty-Fourth Asilomar Conference on Signals, Systems and Computers (Cat. No. 00CH37154)*, volume 1, pages 25–29. IEEE, 2000.

[19] C. Fulton, J. L. Salazar, Y. Zhang, G. Zhang, R. Kelly, J. Meier, M. McCord, D. Schmidt, A. D. Byrd, L. M. Bhowmik, S. Karimkashi, D. S. Zrnic, R. J. Doviak, A. Zahrai, M. Yeary, and R. D. Palmer. Cylindrical polarimetric phased array radar: Beamforming and calibration for weather applications. *IEEE Transactions on Geoscience and Remote Sensing*, 55(5):2827–2841, May 2017.

[20] X. Gao, O. Edfors, F. Rusek, and F. Tufvesson. Massive MIMO performance evaluation based on measured propagation data. *IEEE Transactions on Wireless Communications*, 14(7):3899–3911, 2015.

[21] R. L. Haupt and H. Southall. Experimental adaptive cylindrical array. In *1999 IEEE Aerospace Conference. Proceedings (Cat. No. 99TH8403)*, volume 3, pages 291–296. IEEE, 1999.

[22] W. Heberling and S. J. Frasier. Evaluation of phased-array weather-radar polarimetry at X-band. In *2018 IEEE Radar Conference (RadarConf18)*, pages 0851–0855. IEEE, 2018.

[23] J. Herd, S. Duffy, D. Carlson, M. Weber, G. Brigham, C. Weigand, and D. Cursio. Low cost multifunction phased array radar concept. In *2010 IEEE International Symposium on Phased Array Systems and Technology*, pages 457–460. IEEE, 2010.

[24] M. Herndon and M. Yeary. Real-time FPGA-based digital predistortion for improved amplifier performance in next generation phased arrays. In *2022 IEEE International Radar Conference (RADAR)*, pages 798–803, 2022.

[25] M. Herndon, M. Yeary, and R. Palmer. Studies of front-end distortion characterization via mutual coupling measurements in phased array systems. In *2020 IEEE International Radar Conference (RADAR)*, pages 798–803, 2020.

[26] M. M. Herndon and M. B. Yeary. Calibration of the cylindrical polarimetric phased array radar via GPS-disciplined bistatic pattern measurement. *IEEE Transactions on Aerospace and Electronic Systems*, 58(2):1299–1315, 2022.

[27] G. Hinton, D. Sager, M. Upton, D. Boggs, D. Carmean, A. Kyker, and P. Roussel. The microarchitecture of the Pentium® 4 processor. In *Intel technology journal*. Citeseer, 2001.

[28] K. Hondl and M. Weber. NOAA's meteorological phased array radar research program. In *2019 IEEE International Symposium on Phased Array System & Technology (PAST)*, pages 1–6. IEEE, 2019.

[29] L. Infante, S. Mosca, and G. Pellegrini. A beam synthesis procedure for matrix-fed cylindrical antenna arrays. In *2016 IEEE International Symposium on Phased Array Systems and Technology (PAST)*, pages 1–5. IEEE, 2016.

[30] P. Ioannides and C. A. Balanis. Uniform circular arrays for smart antennas. *IEEE Antennas and Propagation Magazine*, 47(4):192–206, 2005.

[31] B. Isom, R. Palmer, R. Kelley, J. Meier, D. Bodine, M. Yeary, B.-L. Cheong, Y. Zhang, T.-Y. Yu, and M. I. Biggerstaff. The atmospheric imaging radar: Simultaneous volumetric observations using a phased array weather radar. *Journal of Atmospheric and Oceanic Technology*, 30(4):655–675, 2013.

[32] S. Karimkashi and G. Zhang. A dual-polarized series-fed microstrip antenna array with very high polarization purity for weather measurements. *IEEE Transactions on Antennas and Propagation*, 61(10):5315–5319, 2013.

[33] E. Kiuchi and I. Ueda. Tactical cylindrical active phased array radar. In *Proceedings of International Symposium on Phased Array Systems and Technology*, pages 222–225. IEEE, 1996.

[34] A. Klilou, S. Belkouch, P. Elleaume, P. Le Gall, F. Bourzeix, and M. M. Hassani. Real-time parallel implementation of pulse-Doppler radar signal processing chain on a massively parallel machine based on multi-core DSP and serial rapidio interconnect. *EURASIP Journal on Advances in Signal Processing*, 2014(1):161, 2014.

[35] C. Koukourlis, G. Kyriacou, S. Mavrides, S. Diamantis, K. T. Spyridakis, J. Sahalos, G. Stratakos, P. Tsenes, and N. Uzunoglou. Design and development of an active printed cylindrical antenna array for radar applications. *Union Radio-Scientifique Internationale*, (8), 2002.

[36] J. Lake, M. Yeary, and R. Palmer. Real-time digital equalization to enhance element-level digital beamforming. In *2019 IEEE Radar Conference (RadarConf)*, pages 1–6. IEEE, 2019.

[37] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta. Massive MIMO for next generation wireless systems. *IEEE Communications Magazine*, 52(2):186–195, 2014.

[38] G. G. Lema, G. T. Tesfamariam, and M. I. Mohammed. A novel elliptical-cylindrical antenna array for radar applications. *IEEE Transactions on Antennas and Propagation*, 64(5):1681–1688, 2016.

[39] Z. Li and G. Zhang. Similarities and differences in clutter detection between electronic scans and mechanical scans with a polarimetric-phased array radar. *IEEE Transactions on Geoscience and Remote Sensing*, pages 1–9, 2021.

[40] Z. Li, G. Zhang, M. H. Golbon-Haghighi, H. Saeidi-Manesh, M. Herndon, and H. Pan. Initial observations with electronic and mechanical scans using a cylindrical polarimetric phased array radar. *IEEE Geoscience and Remote Sensing Letters*, 18(2):271–275, 2021.

[41] U. Meyer-Baese and U. Meyer-Baese. *Digital Signal Processing with Field Programmable Gate Arrays*, volume 65. Springer, 2007.

[42] D. R. Morgan, Z. Ma, J. Kim, M. G. Zierdt, and J. Pastalan. A generalized memory polynomial model for digital predistortion of RF power amplifiers. volume 54, pages 3852–3860, Oct 2006.

[43] D. R. Morgan, Z. Ma, J. Kim, M. G. Zierdt, and J. Pastalan. A generalized memory polynomial model for digital predistortion of RF power amplifiers. volume 54, pages 3852–3860. IEEE, 2006.

[44] S. Narayanan. Transistor distortion analysis using Volterra series representation. volume 46, pages 991–1024. Wiley Online Library, 1967.

[45] P. O'Leary, M. Harker, and R. Neumayr. Savitzky-Golay smoothing for multivariate cyclic measurement data. In *2010 IEEE Instrumentation Measurement Technology Conference Proceedings*, pages 1585–1590, May 2010.

[46] R. D. Palmer, C. J. Fulton, J. Salazar, H. Sigmarsson, and M. Yeary. The "Horus" radar—an all-digital polarimetric phased array radar for multimission surveillance. In *99th American Meteorological Society Annual Meeting*. AMS, 2019.

[47] R. Penrose. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413. Cambridge University Press, 1955.

[48] C. Pui, M. Trinkle, and B. Ng. Aircraft detection experimental results for GPS bistatic radar using phased-array receiver. 2013.

[49] C. Y. Pui and M. Trinkle. GPS bistatic radar using phased-array technique for aircraft detection. In *2013 International Conference on Radar*, pages 274–279. IEEE, 2013.

[50] J. L. Salazar, T.-Y. Yu, M. McCord, J. Diaz, J. A. Ortiz, C. Fulton, M. Yeary, R. Palmer, B.-L. Cheong, H. Bluestein, et al. An ultra-fast scan C-band polarimetric atmospheric imaging radar (PAIR). In *2019 IEEE International Symposium on Phased Array System & Technology (PAST)*, pages 1–5. IEEE, 2019.

[51] J. Sandenbergh, M. Inggs, and W. Al-Ashwal. Evaluation of coherent netted radar carrier stability while synchronised with GPS-disciplined oscillators. In *2011 IEEE RadarCon (RADAR)*, pages 1100–1105. IEEE, 2011.

[52] A. Savitzky and M. J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964.

[53] M. Schetzen. *The Volterra and Wiener Theories of Nonlinear Systems*. Wiley, 1980.

[54] J. E. Stailey and K. D. Hondl. Multifunction phased array radar for aircraft and weather surveillance. *Proceedings of the IEEE*, 104(3):649–659, 2016.

[55] H. Steyskal and J. S. Herd. Mutual coupling compensation in small array antennas. volume 38, pages 1971–1975. IEEE, 1990.

[56] A. Stumme, W. M. Dorsey, J. Valenzi, and O. Kilic. Additively-manufactured cylindrical array with snap-fit connector integration. In *2019 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting*, pages 85–86. IEEE, 2019.

[57] P. Tirumalai, M. Lee, and M. Schlansker. Parallelization of loops with exits on pipelined architectures. 1990.

[58] G. Ungureanu, T. Sundström, A. Åhlander, I. Sander, and I. Söderquist. Formal design, co-simulation and validation of a radar signal processing system. In *2019 Forum for Specification and Design Languages (FDL)*, pages 1–8. IEEE, 2019.

[59] K. L. Virga and H. Zhang. Spatial beamformer weighting sets for circular array STAP. In *Proceedings 2000 IEEE International Conference on Phased Array Systems and Technology (Cat. No. 00TH8510)*, pages 561–564. IEEE, 2000.

[60] X. Wang, T. Yu, H. Hsiao, and J. Anderson. Double-pumping the interconnect for area reduction in coarse-grained reconfigurable arrays. In *2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 242–249, 2021.

[61] D. S. Watkins. *Fundamentals of Matrix Computations*, volume 64. John Wiley & Sons, 2004.

[62] M. Yeary, G. Crain, A. Zahrai, C. D. Curtis, J. Meier, R. Kelley, I. R. Ivic, R. D. Palmer, R. J. Doviak, G. Zhang, and T.-Y. Yu. Multichannel receiver design, instrumentation, and first results at the National Weather Radar Testbed. *IEEE Transactions on Instrumentation and Measurement*, 61(7):2022–2033, 2012.

[63] M. Yeary, R. Palmer, C. Fulton, J. Salazar, and H. Sigmarsson. Recent advances on an S-band all-digital mobile phased array radar. In *2019 IEEE International Symposium on Phased Array System & Technology (PAST)*, pages 1–5. IEEE, 2019.

[64] E. Yildirim and E. Ercil. Development of an L-band cylindrical phased array. In *2013 IEEE International Symposium on Phased Array Systems and Technology*, pages 746–751. IEEE, 2013.

[65] G. Zhang, R. J. Doviak, D. S. Zrnić, R. Palmer, L. Lei, and Y. Al-Rashid. Polarimetric phased-array radar for weather measurement: A planar or cylindrical configuration? *Journal of Atmospheric and Oceanic Technology*, 28(1):63–73, 2011.