Analog Signal Processor for Adaptive Antenna Arrays

by

Mircea Hossu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2007

Abstract

An analog circuit for beamforming in a mobile Ku band satellite TV antenna array has been implemented. The circuit performs continuous-time gradient descent using simultaneous perturbation gradient estimation. Simulations were performed using Agilent ADS circuit simulator. Field tests were performed in a realistic scenario using a satellite signal. The results were comparable to the simulation predictions and to results obtained using a digital implementation of a similar stochastic approximation algorithm.

Acknowledgements

I would like to thank the iMARS team at the University of Waterloo for their support and guidance, especially my supervisor, Professors Safieddin Safavi-Naeini and Hamid Jamali. I would also like to express my deep appreciation to Intelwaves Technologies Ltd. for providing the adaptive antenna array hardware which made it possible to verify the theory in a real application environment.

Dedication

This thesis is dedicated to my family.

Contents

1 Introduction **1**

1.1 Notation conventions 2

1.2 Antenna array basics 3

1.3 Beamforming optimization problem 4

1.4 Beamforming architecture 4

1.4.1 Linear beamforming 5

1.4.2 Components 7

1.5 Control algorithm 10

1.6 Related work 10

2 Optimization by stochastic approximation **12**

2.1 Deterministic steepest descent 12

2.2 Optimization using stochastic approximation 13

2.3 Stochastic gradient form of stochastic approximation 13

2.3.1 Least mean squares (LMS) 15

2.4 Gradient-free stochastic approximation 16

2.4.1 Finite difference stochastic approximation (FDSA) 16

2.4.2 Simultaneous perturbation stochastic approximation (SPSA) 17

2.4.3 SPSA with single function evaluation 17

2.5 Continuous-time gradient descent by simultaneous perturbation 18

3 Analysis and design of smart antenna system **21**

3.1 Test function 22

3.2 Antenna function 22

3.3 Analog noise sources 26

3.4 Gradient estimation non-idealities 30

3.4.1 Target function transient response 30

3.4.2 Target function unwanted noise 31

3.5 Frequency domains 32

3.6 Phase wrap-around 33

3.7 Simulation results 33

4 Implementation and results **40**

4.1 Implementation 40

4.2 Verification measurements 43

4.3 Experimental results 43

4.3.1 Received signal strength vs pointing direction 46

4.3.2 Transient tests 53

5 Conclusions and future work **59**

Bibliography **60**

A Simulation schematics **62**

B Optimizer implementation schematics **65**

List of Figures

1.1 Conventional vs. smart antennas 2
1.2 Beamforming architectures 6
1.3 Proposed beamforming architecture 7
1.4 Linear beamforming 7
1.5 Antenna sub-array layout, pattern and array geometry 8
1.6 Phase shifter transfer characteristics vs control voltage (random sample set) . . . 9

2.1 Continuous-time gradient descent by simultaneous perturbation 20

3.1 Smart antenna system components 23
3.2 Circuit realizations 24
3.3 Test function and a derivative 25
3.4 Antenna function components 27
3.5 Antenna function and a derivative 28
3.6 Noise source 28
3.7 Noise signal characteristics 29
3.8 Gradient estimation with non-idealities 31
3.9 Correlation factor vs. phase shift 32
3.10 Frequency bands in the optimizer circuit (magnitude in dB) 34
3.11 Phase wrap-around in the positive direction 35
3.12 Simulation of test function transient response without measurement noise 37
3.13 Simulation of test function transient response with measurement noise 38
3.14 Simulation of antenna function transient response 39

4.1 Optimizer board block diagram 42
4.2 Complete system photograph 44
4.3 RF detector step response and inherent noise 45

4.4 Generated dither noise . . . 45
4.5 Signal strength vs. pointing direction with algorithm off . . . 47
4.6 Power vs direction with fixed phases . . . 48
4.7 Signal strength vs. pointing direction with algorithm on . . . 49
4.8 Power vs direction with analog optimizer . . . 50
4.9 Power vs direction with digital optimizer . . . 51
4.10 Multiple beamforming solutions . . . 52
4.11 Power vs direction with analog optimizer, multiple trials . . . 52
4.12 Power vs direction with digital optimizer, multiple trials . . . 53
4.13 Test function transient response to fixed function step . . . 54
4.14 Test function transient response to random function step . . . 55
4.15 Test function transient response from random initial state . . . 55
4.16 Antenna function transient response from random initial state . . . 56
4.17 Antenna function transient response from random initial state . . . 57
4.18 Antenna step reorientation . . . 57
4.19 Antenna function transient response to motor $6.3°$ step . . . 58

A.1 Simulation schematics: single optimizer unit . . . 63
A.2 Simulation schematics: top level . . . 64

B.1 Implementation schematics page 1/9 . . . 66
B.2 Implementation schematics page 2/9 . . . 67
B.3 Implementation schematics page 3/9 . . . 68
B.4 Implementation schematics page 4/9 . . . 69
B.5 Implementation schematics page 5/9 . . . 70
B.6 Implementation schematics page 6/9 . . . 71
B.7 Implementation schematics page 7/9 . . . 72
B.8 Implementation schematics page 8/9 . . . 73
B.9 Implementation schematics page 9/9 . . . 74

Chapter 1

Introduction

Reception of digital satellite television usually begins with a dish antenna. Pointing a satellite dish requires a precise alignment of the high-gain direction of the antenna with the direction of the satellite. This is a very difficult operation because the direction of high-gain is very narrow, in the order of a few degrees. Fortunately, once pointed, no further adjustments are required, thanks to the fact that the satellite is in geostationary orbit.

Mobile satellite TV applications complicate the problem. Some applications require that the mobile receiver be operated only while it is stationary. This demands a fast automated mechanical antenna pointing system which is operated every time the receiver moves to a new location. Other applications require that the receiver be operated continuously while it is moving. This necessitates a system offering precise real-time tracking capability. It is not a trivial task to design a mechanical tracking system which rotates an object having a high moment of inertia (the antenna) with absolute errors in the order of one degree. For one thing, the mechanical system must be extremely fast, requiring high torque and high power actuators.

Fortunately, unlike the satellite dish, there exist antennas whose reception *pattern* can be modified. These are antennas built using a multitude of smaller antennas and are called *adaptive antenna arrays*. Unlike mechanical positioning of the antenna, changing the antenna pattern is an electrical process which does not involve any type of mechanical inertia, resulting in an almost instantaneous response. An antenna array still needs to be mechanically pointed in the right direction, but the required precision is significantly relaxed. Therefore a possible control strategy consists of a moderate precision mechanical pointing system together with an antenna *beamforming* algorithm.

The antenna array with the beamforming algorithm together form a *smart antenna*. Figure 1.1 illustrates the difference between a conventional antenna and a smart antenna. The conventional antenna signal strength quickly drops as the target moves away from the center of the main *beam*. On the other hand, a smart antenna is able to move the main beam to track the target. In this application, the smart antenna provides an artificial substitute for a wide beam, high gain conventional antenna. A conventional antenna with these properties may be impossible to realize given the geometric constraints.

This work focuses on the antenna beamforming algorithm which maximizes received signal power from an antenna array. Typically beamforming algorithms are implemented in software.

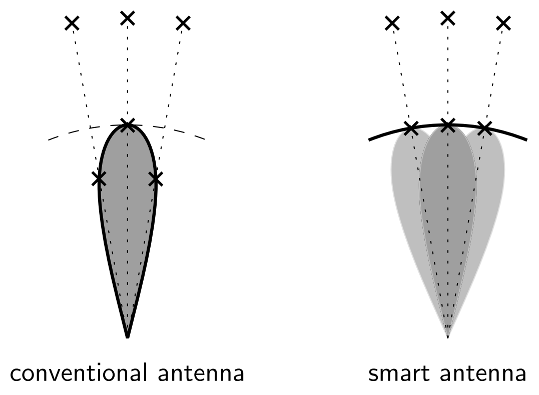


Figure 1.1: Conventional vs. smart antennas

They require additional digital-analog conversion devices as well as a digital signal processor which executes the algorithm. In the interest of lower system complexity, lower power consumption and lower cost the algorithm which will be used is implemented as an analog circuit. Normally the choice of using an antenna array in a certain application has a strong impact on the architecture of the complete receiver. By performing the beamforming algorithm in a circuit operating as independently as possible and close to the antenna, it is envisioned that the resulting smart antenna can be used almost like a conventional antenna in applications similar to this one. Although the prototype implementation is realized using discrete components, circuits of this and greater complexity are routinely realized using VLSI, which is very attractive for mass production.

The remainder of this chapter describes the beamforming problem and surveys related work. The antenna beamforming problem will be shown to be an optimization problem. All work in this thesis considers only one class of optimization algorithms called *stochastic approximation* (SA). Chapter 2 provides a background of optimization using stochastic approximation. Chapter 3 presents the theory and design of an analog circuit which performs gradient descent optimization. Various simulations are presented. Chapter 4 describes an actual implementation of a four-variable optimizing analog circuit. The circuit is implemented using readily available discrete components. Test results obtained outdoors with a satellite signal received using a four-element antenna array controlled by this circuit are presented. Chapter 5 summarizes the results and discusses future work.

1.1 Notation conventions

The following are the general notation conventions used throughout this thesis. Scalar variables are represented in normal weight font, e.g. x. Vector variables are represented by a boldface font, e.g. $\boldsymbol{x}$. Similarly, scalar and vector valued functions are represented as $f(.)$ and $\boldsymbol{f}(.)$ respectively. All vectors are column vectors. The vector or matrix transpose is represented by the subscript ^{T}, e.g. $\boldsymbol{x}^T$. A single vector component is denoted by a zero-based subscript, e.g. x_i. Subscripts are used for other purposes as well, for example to denote a sequence of values. Such cases will be documented independently and furthermore the use should be clear from the context. In the absence of an operator between two variables, a scalar or matrix multiplication operation is assumed, e.g $\boldsymbol{x}^T\boldsymbol{\theta}$.

1.2 Antenna array basics

The following is a brief overview of antenna array theory as described by [1]. It is described from the point of view of signal transmission, reception being identical because of reciprocity. The far-field *electric field* of a single transmitting antenna element can be expressed as follows:

$$\boldsymbol{E_e}(r,\theta,\phi) = \frac{e^{-jkr}}{r}\boldsymbol{f_e}(\theta,\phi)$$

The electric field is a vector valued function describing the field strength and direction (*polarization*) evaluated at a point specified by vector $\boldsymbol{r}$, whose spherical components are (r,θ,ϕ), when an antenna located at the origin is excited with a sinusoidal current; $k = \omega\sqrt{\mu_0\epsilon_0} = 2\pi/\lambda_0$ is the wave *propagation constant*; λ_0 is the free-space wavelength. The factor before $\boldsymbol{f_e}(.)$ describes a diminishing signal level and a phase shift as one moves away from the antenna. Given N such identical antennas, indexed by i, each being excited by the same sinusoidal signal multiplied by a complex constant I_i the total electric field can be found by superposition:

$$\boldsymbol{E}(r,\theta,\phi) = \sum_{i=1}^{N} I_i\frac{e^{-jkr_i}}{r_i}\boldsymbol{f_e}(\theta_i,\phi_i)$$

If one is observing the pattern from the array far-field ($r \geq \frac{2l^2}{\lambda_0}$) where l is the largest dimension of the antenna array, the whole array will resemble a point and $r_i \approx r$, $\theta_i \approx \theta$, $\phi_i \approx \phi$ (the *paraxial approximation*). The approximation $r_i \approx r$ cannot be used in the phase shift term e^{-jkr_i} because even a small difference in r_i (fraction of wavelength) results in a non-ignorable phase shift — indeed this phase shift is the reason why phased arrays work. Therefore, applying these valid approximations one can express the total electric field as follows:

$$\boldsymbol{E}(r,\theta,\phi) = \frac{1}{r}\boldsymbol{f_e}(\theta,\phi)\sum_{i=1}^{N} I_i e^{-jkr_i}$$

Based on a simple geometrical observation we can say $r_i = r + \boldsymbol{p_i}\cdot\boldsymbol{r}$, where $\boldsymbol{p_i}$ is the position vector of each element. Using this and the fact that phase is relative — absolute phase is not important — we get the final expression for the far-field electric field of the array:

$$\boldsymbol{E}(r,\theta,\phi) = \frac{1}{r}\boldsymbol{f_e}(\theta,\phi)\sum_{i=1}^{N} I_i e^{-jk\boldsymbol{p_i}\cdot\boldsymbol{r}}$$

The *radiation intensity* is

$$\begin{aligned}U(\theta,\phi) &= r^2\frac{|\boldsymbol{E}(\theta,\phi)|^2}{2Z_0}\\ U(\theta,\phi) &= U_e(\theta,\phi)\left|\sum_{i=1}^{N} I_i e^{-jk\boldsymbol{p_i}\cdot\boldsymbol{r}}\right|^2\end{aligned}$$

$U_e(.)$ is the radiation intensity of a single element and Z_0 is the *characteristic impedance* of free space. *Antenna directivity* and *antenna gain* are two additional measures which describe the antenna radiation as a function of direction; both are related to radiation intensity by fixed constant factors. It can be easily seen that the radiation intensity in a certain direction can be adjusted by adjusting the complex coefficients I_i.

1.3 Beamforming optimization problem

A beamforming algorithm will adjust I_i in order to *maximize/minimize* a certain *performance measure*. The choice of performance measure depends on the application (environment, signal modulation, etc.). For example, in many applications one has a desired signal arriving from a certain direction and one or more interfering signals arriving from other directions. An intelligent beamforming algorithm can adjust I_i to achieve maximum gain in the desired direction and, usually more importantly, minimum gain in the interference directions. In applications which use a constant amplitude signal modulation scheme, such as Quadrature Phase-Shift Keying (QPSK), a useful performance measure is the *constant modulus* (CM) criterion ([2]):

$$J = \frac{1}{4}E\left\{\left||y(k)|^2 - 1\right|^2\right\}$$

Here $y(k)$ is the total received signal at time k and $E(\cdot)$ is the expectation operation over all k. When J is minimized the desired signal will be amplified, while the interfering signals will be attenuated.

In our current satellite application interference is not an issue. Maximizing the following simple performance measure is sufficient:

$$J = E\left\{|y(k)|\right\}$$

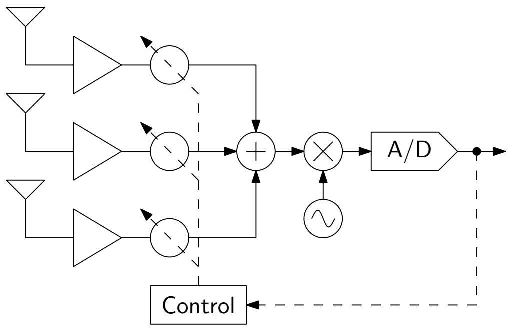
1.4 Beamforming architecture

There are multiple ways to implement an antenna beamforming system. Figure 1.2 shows four important beamforming architectures as they are presented in [3]. *Microwave beamforming* (Figure 1.2 (a)) performs signal weighting and summation using RF components (phase shifters, amplifiers, and a power combiner). The algorithm is implemented in the digital domain; it uses as input the single combined baseband signal and adjusts the RF weights. *Digital beamforming* (Figure 1.2 (b)) keeps all received signals separate to baseband, at which point they are combined in the digital domain. This scheme has the advantage that all signals are available to the controller which is required for some beamforming algorithms. One obvious disadvantage is the complete replication of hardware for each additional antenna. *Local beamforming* (Figure 1.2 (c)) is similar to microwave beamforming but the signal weighting is achieved by modifying the local oscillator. *Aerial beamforming* (Figure 1.2 (d)) is an interesting approach in which the signals received by a number secondary antennas add through mutual coupling with the signal received by a single main antenna. The relative amplitudes and phases of the coupled signals can be controlled by varying the reactance of the secondary antennas. A particular antenna implementation is the *electronically steerable passive array radiator* (ESPAR) antenna ([4]). This method requires the absolute minimum amount of hardware, the same amount of hardware necessary when using a single conventional antenna.

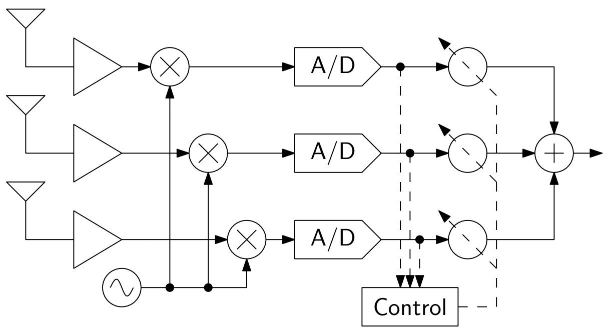
The architecture used in this work is shown in Figure 1.3. It is a microwave beamforming architecture similar to Figure 1.2 (a) except that the control algorithm is implemented completely in the analog domain using only the combined RF signal as an input. The circuit to the right of the power combiner is grayed out to show that it does not play any part in the algorithm; in fact the complete smart antenna system is tested using a consumer digital satellite TV receiver which expects a standard satellite dish signal as an input. In practical applications the algorithm will require a few simple control inputs (e.g. reset) which the receiver must control; the idea is to perform as much processing as possible without the participation of the receiver.

1.4.1 Linear beamforming

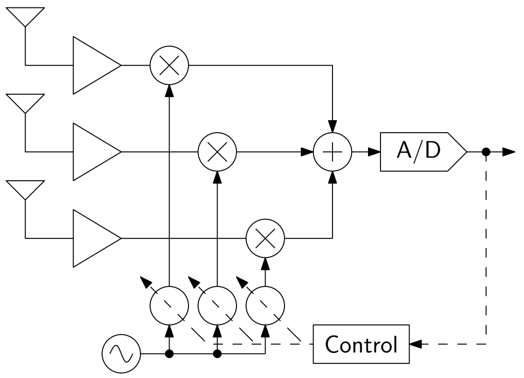
It is interesting to note that antenna beamforming of narrowband signals can be achieved by a linear combination of $2N$ signals where N is the number of antennas, as explained in [5]. It is a simple exercise to show that a phase shift and gain applied to a signal $cos(\omega t)$ can be achieved by the linear combination $w_1 cos(\omega t) + w_2 cos(\omega t - 90°)$ where w_1 and w_2 are real weights (only gain). If we construct the $2N$ column vector $\boldsymbol{x}$ by taking all N original signals and all N signals obtained by delaying each original signal by $90°$, and we construct the $2N$ column vector $\boldsymbol{w}$ from weights, the beamformer output is $y = \boldsymbol{x}^T \boldsymbol{w}$ (Figure 1.4). This type of beamforming architecture is useful because it allows one to apply very popular algorithms which assume a linear model, such as the *least mean squares* (*LMS*) algorithm. However, additional cost is incurred by the extra $90°$ phase shifts and the $2N$ amplifiers, as well as the doubling of dimensionality from N to $2N$. In addition, most algorithms in this class require the full vector $\boldsymbol{x}$ for finding the optimum solution. The particular beamforming architecture which was the target of this work (Figure 1.3), being already designed and manufactured, was considered fixed. This made it impossible to apply algorithms which assume a linear model. Therefore, non-linear optimization algorithms were pursued. Note that non-linear optimization algorithms are more generic and are also effective on the "easier" subclass of linear-type problems ([6]). For all these reasons the focus will be non-linear optimization algorithms.



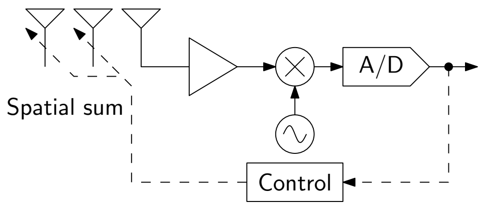
(a) Microwave beamforming



(b) Digital beamforming



(c) Local beamforming



(d) Aerial beamforming

Figure 1.2: Beamforming architectures

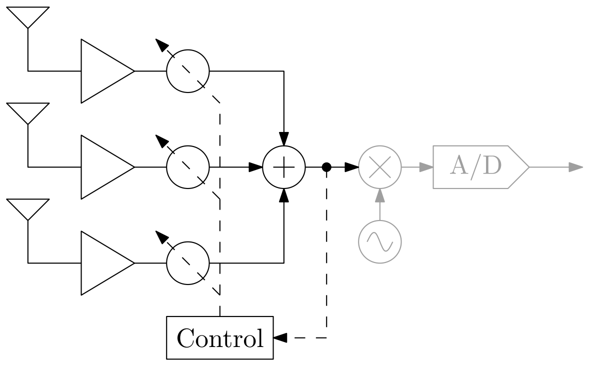


Figure 1.3: Proposed beamforming architecture

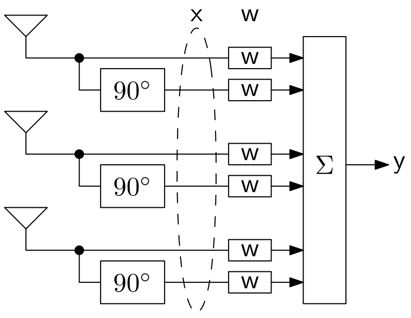
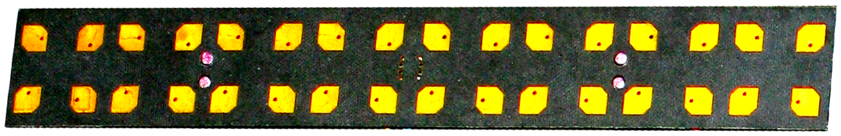


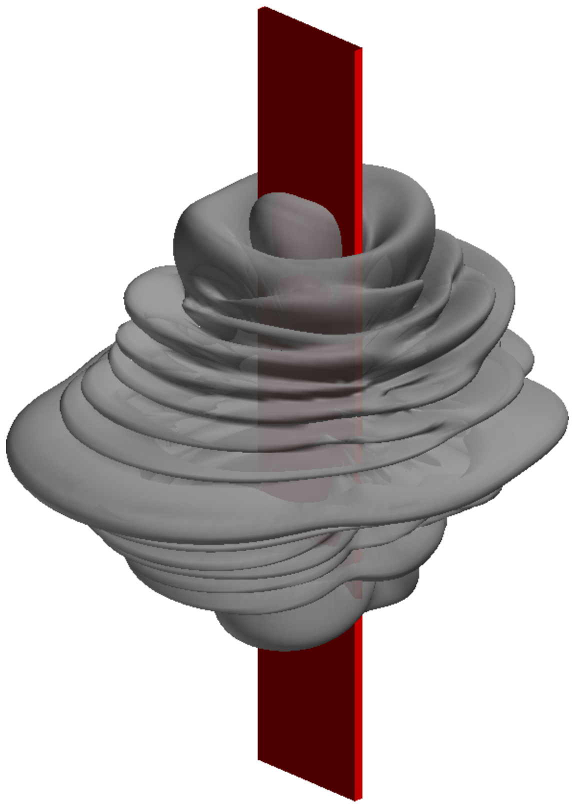
Figure 1.4: Linear beamforming

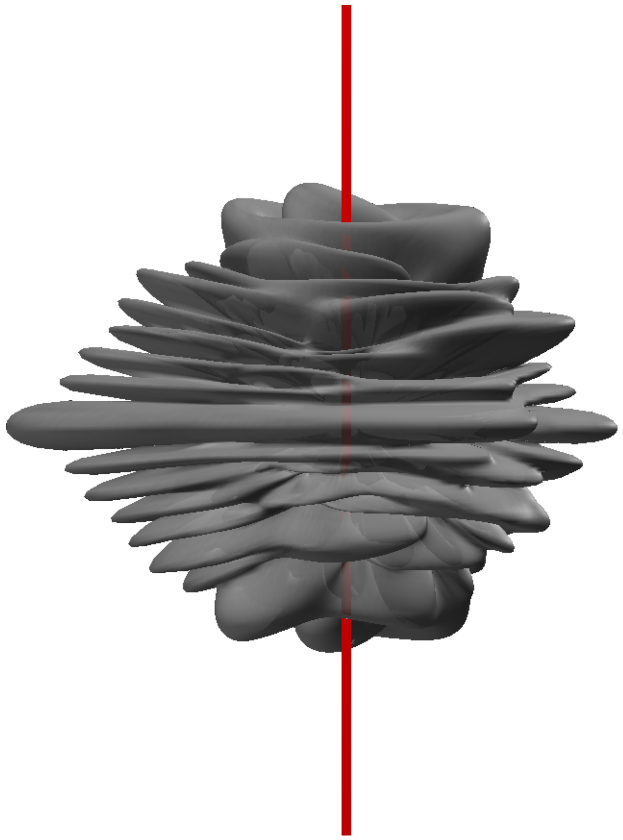
1.4.2 Components

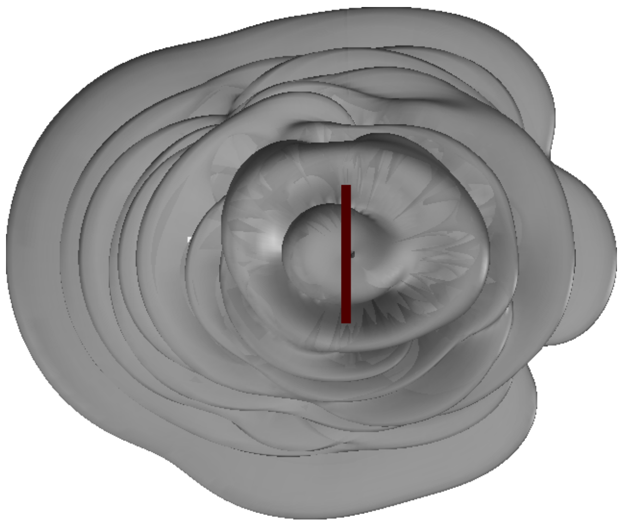
This subsection gives a brief description of the characteristics of some of the more important components in the antenna array for which the algorithm is ultimately targeted. The antenna array is composed of four identical sub-arrays, each in turn being composed of dozens of microstrip patch antenna elements connected together by a fixed feed network. The sub-array layout is shown in Figure Figure 1.5 (a). The lab measured radiation intensity pattern (in dB) of one sub-array is presented in Figure 1.5 (b)–(d). The thin rectangular structure represents the sub-array geometry. Note that the sub-array is fixed and for all practical points of view it is just a normal (non-adaptive) antenna. Figure 1.5 (e) shows how the four sub-arrays are oriented in the antenna array.

The phase shifter is a series device which uses varactors as delay elements. The phase vs. control voltage and gain vs. control voltage are presented in Figure 1.6. These are *typical* curves for a random set of phase shifters. For an ideal phase shifter, the phase vs. control voltage would be a straight line with a non-zero slope, while the gain vs. control voltage would be a straight line with a zero slope. Two difficulties can be seen: the curves are not ideal and the curves vary from phase shifter to phase shifter. In addition to the manufacturing variations, there are also temperature and age variations which were not formally characterized.


(a) Sub-array layout


(b) Pattern oblique view


(c) Pattern side view


(d) Pattern top view

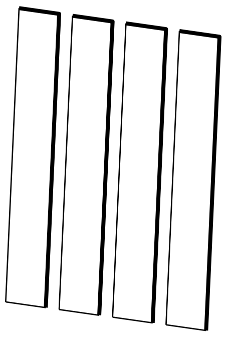
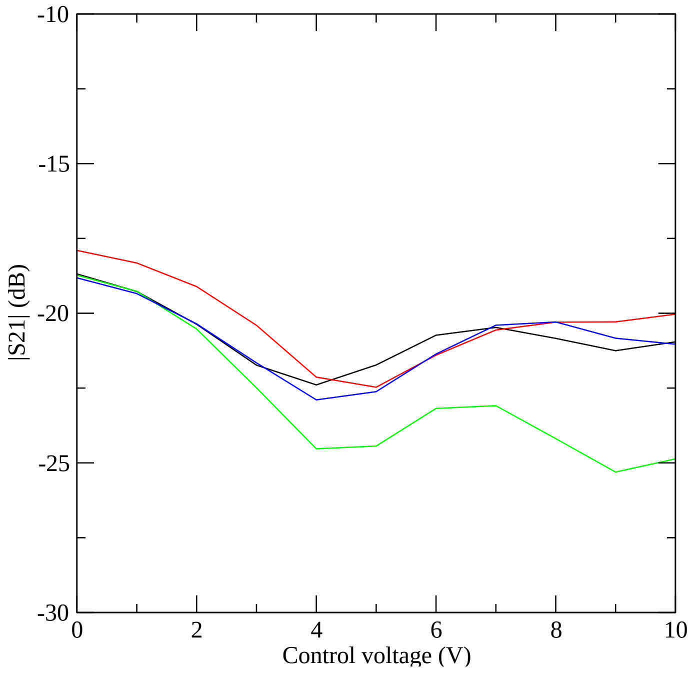
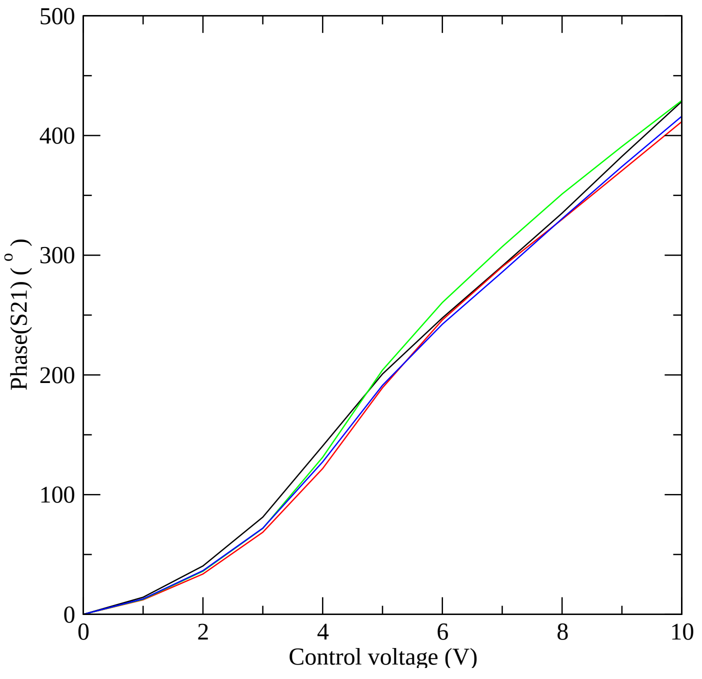

(e) Array geometry

Figure 1.5: Antenna sub-array layout, pattern and array geometry



(a) Magnitude



(b) Phase

Figure 1.6: Phase shifter transfer characteristics vs control voltage (random sample set)

1.5 Control algorithm

The complex characteristics of the antenna elements and the phase shifters which were seen in the previous sub-section would require very complex modeling if one wishes to incorporate these non-ideal characteristics in the control algorithm. In mass-production it might even be completely impractical to follow such an approach. For this reason the algorithms which were pursued are of the *model-free* type. This means that the algorithm is generic in the sense that it is not designed specifically for this particular set of antenna and phase shifters. This not only decouples the problem of RF design from that of algorithm design, but also greatly increases the value of the algorithm by virtue of reusability. For this reason a class of generic optimization algorithms called *stochastic approximation* was considered for this particular problem. Stochastic approximation algorithms have had tremendous success across widely ranging fields from economics to artificial intelligence ([6]).

Some main characteristics of stochastic approximation algorithms are introduced here and discussed in more detail in Chapter 2. The algorithm finds the solution iteratively; i.e. the solution at time t_2 is on average better than the solution at an earlier time t_1. The solution proceeds in the direction of maximum (or minimum) gradient; i.e. the algorithm performs *gradient descent*. The solution most likely converges to a local maximum (or minimum); i.e. the algorithm generally performs *local* as opposed to *global* optimization.

An algorithm which uses gradient descent but has no knowledge of the target function needs to "probe" the function to find out the gradient at the current solution point. Gradient estimation in the presence of noisy target function measurements is the key problem. The performance and efficiency of gradient descent algorithms are determined by the procedure by which the gradient is estimated; this procedure is what distinguishes one gradient descent algorithm from another. *Sequential perturbation* is one way of measuring the gradient of a function. The procedure consists of applying a very small change, the *perturbation* to one of the inputs and measuring the change in output, thus obtaining an estimate of the partial derivative. Repeating this operation once for each input results in the complete set of partial derivatives which make up the gradient. A much more efficient method, called *simultaneous perturbation*, applies all the changes to the inputs simultaneously and observes one single output change. Obtaining a gradient in this manner, while it may intuitively appear crude and inaccurate, turns out to give algorithm convergence rates similar to sequential perturbation while providing tremendous cost savings in terms of target function evaluations. Simultaneous perturbation stochastic approximation algorithms were the main candidates pursued in this work.

1.6 Related work

Model-free optimization algorithms with simultaneous perturbation gradient estimation have been implemented several times. Most of the advances were made in the research area of neural networks. In 1990 [7] was the first to provide a comprehensive discussion of simultaneous perturbation gradient descent for neural network learning. Building upon this, [8] describes a simultaneous perturbation gradient descent algorithm suitable for neural network learning. The author follows through with an analog VLSI single-chip implementation ([9]) containing both the learning algorithm and the neural network. [10] also describes a similar parallel learning algorithm with focus on analog VLSI implementation and provides an outline of such an implementation. [11] describes an analog VLSI single-chip implementation of a similar learning algorithm, but presents the circuit as generic in the sense that any analog function can be optimized. This particular publication ([11]) was the starting point for the implementation of the optimization circuit in the present work. [12] describes the implementation of a mixed analog/digital circuit which performs neural network learning using a stochastic gradient descent algorithm with simultaneous perturbation gradient estimation. In the research area of adaptive filters [13] presents a similar algorithm for tuning an adaptive filter, obtaining benefits over the traditional way of tuning adaptive filters using the LMS algorithm which requires full knowledge of the filter tap outputs.

Many simultaneous perturbation algorithms require pseudo-random uncorrelated noise sources. A very convenient method of implementing multiple uncorrelated analog noise sources was presented in [14]. This method of generating the noise sources was used by [11]. The present work also uses this method of generating noise sources.

The microwave beamforming architecture used in this work has been very well known for decades ([5]). The application of model-free stochastic approximation algorithms to antenna beamforming has also been performed. Some very relevant research is done on the ESPAR antenna. Although this involves aerial beamforming instead of microwave beamforming, the problem is similar because only the combined antenna output is available for optimization. Several different model-free gradient descent algorithms have been tried on the ESPAR antenna. Experimental results using sequential perturbation gradient estimation are presented in [15]. Experimental results using simultaneous perturbation gradient estimation are presented in [16]. In both instances the algorithm was implemented in software running on a digital processor operating on the single baseband combined signal. Nonetheless, results are relevant to this work since the performance of an analog implementation is expected to be similar to that obtained from a digital implementation.

As far as I am aware, there are no published results of an analog circuit implementation of simultaneous perturbation stochastic approximation applied to antenna microwave beamforming. This is the topic of the present work.

Chapter 2

Optimization by stochastic approximation

Stochastic approximation is a class of gradient descent algorithms used for stochastic optimization. The following discussion of optimization is based mostly on [6]. The same notation and terminology is used as much as possible. Optimization problems are either maximization problems or minimization problems. In essence both are the same up to a single sign change and therefore it is sufficient to deal only with one of them. In the present discussion we will refer to the minimization problem.

The two optimization problems of interest are the minimization of a real valued scalar function $L(\boldsymbol{\theta})$ and the root finding problem of solving for $\boldsymbol{\theta}$ in the vector equation $\boldsymbol{g}(\boldsymbol{\theta}) = \mathbf{0}$. The best solution is noted as $\boldsymbol{\theta}^{\star}$. $L(\boldsymbol{\theta})$ is assumed to be a continuous function of $\boldsymbol{\theta}$ at least once differentiable. Furthermore $L(\boldsymbol{\theta})$ has at least one minimum; it may have more minimums, but it is assumed that any minimum found is an acceptable solution to the problem at hand. That is, we are looking only at local optimization algorithms which converge on any local minimum. In the special case that $\boldsymbol{g}(\boldsymbol{\theta})$ is the gradient of $L(\boldsymbol{\theta})$, solving $\boldsymbol{g}(\boldsymbol{\theta}) = \mathbf{0}$ also minimizes $L(\boldsymbol{\theta})$ and the two optimization problems are thus equivalent.

The gradient is defined as a p-column vector composed of partial derivatives with respect to the components of $\boldsymbol{\theta}$:

$$\boldsymbol{g}(\boldsymbol{\theta}) \equiv \frac{\partial L}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial L}{\partial \theta_1} \\ \frac{\partial L}{\partial \theta_2} \\ \vdots \\ \frac{\partial L}{\partial \theta_p} \end{bmatrix}$$

2.1 Deterministic steepest descent

The meaning of "stochastic" in steepest descent stochastic optimization signifies randomness in the measurement of $L(\boldsymbol{\theta})$ or $\boldsymbol{g}(\boldsymbol{\theta})$. In the absence of noise the problem becomes a deterministic optimization problem. One very well known algorithm which can be applied is the *method of steepest descent*, where $\hat{\boldsymbol{\theta}}_k$, the current best solution, is updated as follows starting from an initial guess $\hat{\boldsymbol{\theta}}_0$:

$$\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \boldsymbol{g}(\hat{\boldsymbol{\theta}}_k)$$

The term $a_k > 0$ is a scalar called the step size which determines the convergence rate and the final accuracy. Various criteria exist for choosing the sequence a_k, the simplest being a constant. Notice that every step is exactly in the opposite direction of the gradient. This direction will point closer to a better solution (closer to $\boldsymbol{\theta}^\star$), but depending on the function being minimized, the direction may or may not be directly towards $\boldsymbol{\theta}^\star$. The *Newton-Raphson* algorithm is a related deterministic algorithm which uses the Hessian (second partial derivatives of $L(\boldsymbol{\theta})$) in order to proceed more directly towards $\boldsymbol{\theta}^\star$ for faster convergence at the cost of potential instability ([6]).

2.2 Optimization using stochastic approximation

In the presence of noisy measurements of $L(\boldsymbol{\theta})$ or $\boldsymbol{g}(\boldsymbol{\theta})$ the problem becomes stochastic. The stochastic optimization problem is defined as a root finding problem as follows. Find at least one root $\boldsymbol{\theta}^\star \in \Theta^\star \subseteq \Theta \subseteq \mathbb{R}^p$ to

$$\boldsymbol{g}(\boldsymbol{\theta}) = \mathbf{0}$$

using noisy measurements $\boldsymbol{Y}_k(\hat{\boldsymbol{\theta}}_k)$ of $\boldsymbol{g}(\boldsymbol{\theta})$ where:

$$\boldsymbol{Y}_k(\hat{\boldsymbol{\theta}}_k) = \boldsymbol{g}(\hat{\boldsymbol{\theta}}_k) + \boldsymbol{e}_k(\hat{\boldsymbol{\theta}}_k)$$

$\boldsymbol{\theta}^\star$ is one root in the set $\Theta^\star$ containing all roots, which is a subset of the set Θ containing all possible values of $\boldsymbol{\theta}$, which is a subset of the set $\mathbb{R}^p$ containing all real-valued vectors , p being the dimensionality of $\boldsymbol{\theta}$. $\boldsymbol{e}_k(\hat{\boldsymbol{\theta}}_k)$ is the noise.

The *basic root-finding (Robbins-Monro) SA* algorithm is

$$\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \boldsymbol{Y}_k(\hat{\boldsymbol{\theta}}_k)$$

This is very similar to the deterministic gradient descent algorithm but instead of $\boldsymbol{g}(\hat{\boldsymbol{\theta}}_k)$ one uses the noisy $\boldsymbol{Y}_k(\hat{\boldsymbol{\theta}}_k)$. Simple as the algorithm may look, convergence analysis is far from simple. [6] provides a summary of the comprehensive theory developed by Robbins and Monro. This theory is the analytical basis for all other SA algorithms which were developed later ([6]).

2.3 Stochastic gradient form of stochastic approximation

The previous section described the Robbins-Monro algorithm for finding the roots of a vector-valued function $\boldsymbol{g}(\boldsymbol{\theta})$ which is evaluated in the presence of noise. In the special case that $\boldsymbol{g}(\boldsymbol{\theta})$ is the gradient of a loss function $L(\boldsymbol{\theta})$ the root finding solution is identical to minimizing the loss function. The deterministic loss function is defined as the expected value of a possibly time-varying function $Q(.)$ of $\boldsymbol{\theta}$ and possibly time-varying randomness V as follows:

$$L(\boldsymbol{\theta}) = E\{Q_k(\boldsymbol{\theta}, V_k)\}$$

$\boldsymbol{g}(\boldsymbol{\theta})$ is the gradient of the loss function:

$$\boldsymbol{g}(\boldsymbol{\theta}) = \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}} E\{Q_k(\boldsymbol{\theta}, V_k)\}$$

A very important property which applies under certain conditions (which are often easily met) is the exchange of the derivative and the expectation operation:

$$\frac{\partial}{\partial \boldsymbol{\theta}} E\{Q_k(\boldsymbol{\theta}, V_k)\} = E\left\{\frac{\partial}{\partial \boldsymbol{\theta}} Q_k(\boldsymbol{\theta}, V_k)\right\}$$

The quantity in the brackets on the right side of the equation is called the *stochastic gradient* of the function $Q(\boldsymbol{\theta}, V)$ because it contains the randomness term V. It will be denoted as $\boldsymbol{Y}(\boldsymbol{\theta})$ and it is an unbiased estimate of $\boldsymbol{g}(\boldsymbol{\theta})$:

$$\boldsymbol{Y}(\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}} Q(\boldsymbol{\theta}, V)$$

Now the Robbins-Monro SA algorithm can be applied:

$$\begin{aligned} \hat{\boldsymbol{\theta}}_{k+1} &= \hat{\boldsymbol{\theta}}_k - a_k \left.\frac{\partial}{\partial \boldsymbol{\theta}} Q_k(\boldsymbol{\theta}, V_k)\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_k} \\ \hat{\boldsymbol{\theta}}_{k+1} &= \hat{\boldsymbol{\theta}}_k - a_k \boldsymbol{Y}_k(\hat{\boldsymbol{\theta}}_k) \end{aligned}$$

Example

To understand the type of application in which this algorithm can be applied, consider a real system which produces a measured scalar output z from a known vector input $\boldsymbol{h}$ in the following way:

$$z_k = f(\boldsymbol{h}_k, \boldsymbol{\theta}) + v_k$$

where the closed form expression of $f(.)$ is known, $\boldsymbol{\theta}$ is a set of fixed unknown parameters of $f(.)$ and v_k is a random variable. To find $\boldsymbol{\theta}$ one may form a loss function $L(\boldsymbol{\theta})$ by computing the *mean squared error* (MSE) between z and the output of an ideal model z':

$$\begin{aligned}
z'_k &= f(\boldsymbol{h}_k, \boldsymbol{\theta}) \\
Q(\boldsymbol{\theta}, v_k) &= (z'_k - z_k)^2 \\
Q(\boldsymbol{\theta}, v_k) &= (f(\boldsymbol{h}_k, \boldsymbol{\theta}) - z_k)^2 \\
L(\boldsymbol{\theta}) &= E\left\{Q(\boldsymbol{\theta}, v_k)\right\}
\end{aligned}$$

Then the stochastic gradient may be formed:

$$\begin{aligned}
Y(\boldsymbol{\theta}, v_k) &= \frac{\partial}{\partial \boldsymbol{\theta}} Q(\boldsymbol{\theta}, v_k) \\
Y(\boldsymbol{\theta}, v_k) &= 2(f(\boldsymbol{h}_k, \boldsymbol{\theta}) - z_k) \frac{\partial}{\partial \boldsymbol{\theta}} f(\boldsymbol{h}_k, \boldsymbol{\theta})
\end{aligned}$$

Note that the stochastic gradient has a randomness term by virtue of dependence on z_k. This stochastic gradient may be used in the gradient descent algorithm to provide the update rule for $\hat{\boldsymbol{\theta}}$:

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{k+1} &= \hat{\boldsymbol{\theta}}_k - a_k \boldsymbol{Y}(\hat{\boldsymbol{\theta}}_k) \\
\hat{\boldsymbol{\theta}}_{k+1} &= \hat{\boldsymbol{\theta}}_k - 2a_k (f(\boldsymbol{h}_k, \hat{\boldsymbol{\theta}}_k) - z_k) \left. \frac{\partial}{\partial \boldsymbol{\theta}} f(\boldsymbol{h}_k, \boldsymbol{\theta}) \right|_{\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}}
\end{aligned}$$

2.3.1 Least mean squares (LMS)

The *least mean squares* (LMS) is a very popular special case of the previous example which applies to *linear models*. Consider a system which generates a measured scalar output z_k from a known scalar input $\boldsymbol{h}_k$ according to the following formula:

$$z_k = \boldsymbol{h}_k^T \boldsymbol{\theta} + v_k$$

That is, the output is a noisy measurement of a linear combination of the input and a parameter vector $\boldsymbol{\theta}$. If one substitutes $f(\boldsymbol{h}_k, \boldsymbol{\theta}) = \boldsymbol{h}_k^T \boldsymbol{\theta}$ in the previous example one obtains the LMS algorithm:

$$\begin{aligned}
Y(\boldsymbol{\theta}, v_k) &= 2(f(\boldsymbol{h}_k, \boldsymbol{\theta}) - z_k) \frac{\partial}{\partial \boldsymbol{\theta}} f(\boldsymbol{h}_k, \boldsymbol{\theta}) \\
Y(\boldsymbol{\theta}, v_k) &= 2\boldsymbol{h}_k^T (\boldsymbol{h}_k^T \boldsymbol{\theta} - z_k) \\
\hat{\boldsymbol{\theta}}_{k+1} &= \hat{\boldsymbol{\theta}}_k - a_k \boldsymbol{Y}(\hat{\boldsymbol{\theta}}_k) \\
\hat{\boldsymbol{\theta}}_{k+1} &= \hat{\boldsymbol{\theta}}_k - 2a_k \boldsymbol{h}_k^T (\boldsymbol{h}_k^T \hat{\boldsymbol{\theta}}_k - z_k)
\end{aligned}$$

2.4 Gradient-free stochastic approximation

In Section 2.3 the gradient of the loss function is available, albeit containing some randomness, hence the name stochastic gradient. There are many problems where the gradient is not specifically available. Furthermore, even if the gradient of the loss function is obtainable, the mathematical manipulation and further software implementation might be prohibitively complex ([6]). For example, in the present work, it would be very time consuming and difficult to incorporate all of the characteristics of the antenna and RF circuit into a formula. In addition, in our beamforming architecture we do not have the needed access to the individual antenna signals (input vector $\boldsymbol{h}$ in the example of Section 2.3). For these types of problems there are a class of techniques for obtaining a *gradient estimate* using only noisy measurements of the loss function. The familiar SA algorithm uses the gradient estimate, $\hat{\boldsymbol{g}}(\boldsymbol{\theta})$, in place of the stochastic gradient to obtain the update rule:

$$\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$$

2.4.1 Finite difference stochastic approximation (FDSA)

The one-sided finite difference method involves measuring the function output $y(\hat{\boldsymbol{\theta}})$ and $y(\hat{\boldsymbol{\theta}} + perturbation)$. Only one component of $\hat{\boldsymbol{\theta}}$ is perturbed for every measurement, thus requiring p function measurements for each gradient estimate:

$$\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k) = \begin{bmatrix} \frac{y\left(\hat{\boldsymbol{\theta}}_k + c_k \boldsymbol{\xi}_1\right) - y\left(\hat{\boldsymbol{\theta}}_k\right)}{c_k} \\ \frac{y\left(\hat{\boldsymbol{\theta}}_k + c_k \boldsymbol{\xi}_2\right) - y\left(\hat{\boldsymbol{\theta}}_k\right)}{c_k} \\ \vdots \\ \frac{y\left(\hat{\boldsymbol{\theta}}_k + c_k \boldsymbol{\xi}_p\right) - y\left(\hat{\boldsymbol{\theta}}_k\right)}{c_k} \end{bmatrix}$$

$\boldsymbol{\xi}_i$ is a column vector having a 1 for element i and 0 everywhere else. c_k is the perturbation magnitude which may be constant.

The two-sided finite difference method involves measuring the function output $y(\hat{\boldsymbol{\theta}}+perturbation)$ and $y(\hat{\boldsymbol{\theta}} - perturbation)$; this involves twice as many measurements as the one-sided method, thus requiring $2p$ function measurements. Below is the two-sided finite difference gradient estimate:

$$\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k) = \begin{bmatrix} \frac{y\left(\hat{\boldsymbol{\theta}}_k + c_k \boldsymbol{\xi}_1\right) - y\left(\hat{\boldsymbol{\theta}}_k - c_k \boldsymbol{\xi}_1\right)}{2c_k} \\ \frac{y\left(\hat{\boldsymbol{\theta}}_k + c_k \boldsymbol{\xi}_2\right) - y\left(\hat{\boldsymbol{\theta}}_k - c_k \boldsymbol{\xi}_2\right)}{2c_k} \\ \vdots \\ \frac{y\left(\hat{\boldsymbol{\theta}}_k + c_k \boldsymbol{\xi}_p\right) - y\left(\hat{\boldsymbol{\theta}}_k - c_k \boldsymbol{\xi}_p\right)}{2c_k} \end{bmatrix}$$

2.4.2 Simultaneous perturbation stochastic approximation (SPSA)

The simultaneous perturbation method obtains a gradient estimate by perturbing all components of $\hat{\boldsymbol{\theta}}$ simultaneously. Only two function measurements are required, an improvement of $2/2p = 1/p$ over the FDSA.

$$\begin{aligned}
\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k) &= \begin{bmatrix} \frac{y\left(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\Delta}_k\right) - y\left(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\Delta}_k\right)}{2c_k\Delta_{k1}} \\ \frac{y\left(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\Delta}_k\right) - y\left(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\Delta}_k\right)}{2c_k\Delta_{k2}} \\ \vdots \\ \frac{y\left(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\Delta}_k\right) - y\left(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\Delta}_k\right)}{2c_k\Delta_{kp}} \end{bmatrix} \\
&= \frac{y\left(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\Delta}_k\right) - y\left(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\Delta}_k\right)}{2c_k}\left[\Delta_{k1}^{-1}, \Delta_{k2}^{-1}, \cdots, \Delta_{kp}^{-1}\right]^T
\end{aligned}$$

Certain probability distribution conditions must be met when randomly generating the components of the p-dimensional perturbation vector $\boldsymbol{\Delta}_k$. One very simple acceptable distribution is the Bernoulli distribution: (-1,+1) outcomes with equal probability.

2.4.3 SPSA with single function evaluation

Obtaining a gradient estimate using a single evaluation of the loss function is of interest because it would seem likely that such an algorithm would be more easily implementable in continuous time. An algorithm which contains several steps (e.g. positive perturbation, negative perturbation) requires time slicing and an associated scheduling controller. On the other hand, an algorithm which requires only one step has no such requirement since it loops through the same state continuously.

[17] introduces a one measurement gradient estimate called *SPSA1* — the “1” indicates *one* measurement — which is calculated as follows:

$$\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k) = \frac{y_k}{c_k}\begin{bmatrix} \Delta_{k1}^{-1} \\ \Delta_{k2}^{-1} \\ \vdots \\ \Delta_{kp}^{-1} \end{bmatrix}$$

$y_k = L(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\Delta}_k) + \epsilon_k$, c_k is a positive scalar, $\boldsymbol{\Delta}_k = [\Delta_{k1}, \Delta_{k2}, \ldots, \Delta_{kp}]^T$ is a vector of zero-mean independent random variables, and ϵ_k is measurement noise. This algorithm is reported to be less efficient than SPSA in terms of the total number of loss function evaluations but for certain class of applications with non-stationary signals the instantaneous gradient measurement may seem attractive ([17]).

[18] also introduces a one measurement gradient descent algorithm called the *time difference simultaneous perturbation* (TDSP) method. The algorithm is as follows:

$$\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - \alpha \hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k) + c\boldsymbol{s}_k$$

$$\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k) = \begin{bmatrix} sat\left(\frac{y_k - y_{k-1}}{c\boldsymbol{s}_{k-1,1}}\right) \\ sat\left(\frac{y_k - y_{k-1}}{c\boldsymbol{s}_{k-1,2}}\right) \\ \vdots \\ sat\left(\frac{y_k - y_{k-1}}{c\boldsymbol{s}_{k-1,p}}\right) \end{bmatrix}$$

α is a positive gain coefficient, $c > 0$ is the magnitude of the perturbation, $\boldsymbol{s}_k$ is a sign column vector composed of random choice between $(+1, -1)$, $E(\boldsymbol{s}_{k,i}) = 0$, $sat(.)$ is a saturation function which saturates the argument in the range $(\Delta\theta_{min}, \Delta\theta_{max})$, and $\boldsymbol{s}_{k,i}$ denotes the i-th component of the vector $\boldsymbol{s}_k$. The algorithm works by adding random perturbations continuously and obtaining a continuous gradient measure from the time difference of the output function y. [18] concludes that this algorithm converges slower than SPSA and recommends future improvements in convergence speed and a proof of convergence.

2.5 Continuous-time gradient descent by simultaneous perturbation

For reasons already discussed, gradient-free SA algorithms are preferred for the beamforming application. Of these, algorithms such as SPSA1 and TDSP, which require only one loss function evaluation are preferred because of the anticipated lower complexity of an analog implementation.

Until now all algorithms were discrete-time algorithms. [7] discusses the conversion from a discrete-time SA algorithm to continuous time. Alternatively, one may use the already available continuous-time method of estimating gradients as described by [11]. This latter approach was pursued first because results of an analog circuit implementation were already available ([11]). The following discussion refers to this particular algorithm.

Figure 2.1 (a) shows the block diagram of the continuous-time algorithm. Only two weights are shown for clarity. $\boldsymbol{d}$ is a p-dimensional column vector of mutually-orthogonal, zero-mean, small-amplitude *dither* signals (the perturbation). [7] provides an analysis of the optimality of these signals and compares several practical realizations. [19] mentions the possibility of using sinusoidal signals of different frequencies. In [11], and for the remaining of this thesis, these signals are zero-mean, uncorrelated random signals produced as in [14]. The gradient descent operation is straightforward: the discrete-time summation is converted to integration. Similar to the discrete-time case, the bulk of the algorithm is the gradient estimation, which is also what distinguishes one algorithm from another.

[11] provides an analysis of the multidimensional gradient estimation operation. Here we will adopt a description consisting of multiple parallel scalar gradient estimation operations, resembling the *distributed* formulation of [7]. If one disconnects the gradient descent part of the circuit and fixes $\hat{\boldsymbol{\theta}}$ at a constant operating point $\hat{\boldsymbol{\theta}}_0$, the gradient estimation operation may be considered as an *open-loop* operation and the function is reduced to the linear function $f(\boldsymbol{d}) = \boldsymbol{g}(\hat{\boldsymbol{\theta}}_0)^T\boldsymbol{d}+\boldsymbol{C}$. Ignoring the DC term and noting that the gradient is a constant vector, we may write the function as follows: $f(\boldsymbol{d}) = \boldsymbol{G}^T\boldsymbol{d}$. This shows the combined effect of the individual dither sources on the function output.

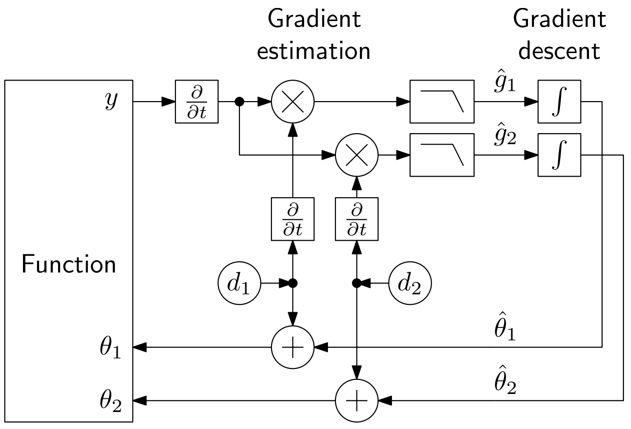
If we look at the i-th branch, given the fact that all dither components are statistically independent, we can combine the effect of all the other $p-1$ dither noise sources $d_j,\ j \neq i$ into a single scalar noise n_i. (To this we can also add any other random effect inherent in the measurement of the loss function.) Now the p-dimensional system can be separated into p parallel scalar systems. The gradient estimation part is shown in 2.1 (b). Below we show that output g is a non-biased estimate of the $G = G_i$. Since all problems are identical, subscript i will be dropped from the following derivation.

$$\begin{aligned}
\frac{\partial(dG+n)}{\partial t}\frac{\partial d}{\partial t} &= \frac{\partial(dG+n)}{\partial d}\left(\frac{\partial d}{\partial t}\right)^2 \\
&= \left(G+\frac{\partial n}{\partial d}\right)\left(\frac{\partial d}{\partial t}\right)^2 \\
&= G\left(\frac{\partial d}{\partial t}\right)^2 + \frac{\partial n}{\partial d}\left(\frac{\partial d}{\partial t}\right)^2 \\
&= G\left(\frac{\partial d}{\partial t}\right)^2 + \frac{\partial n}{\partial t}\frac{\partial d}{\partial t}
\end{aligned}$$

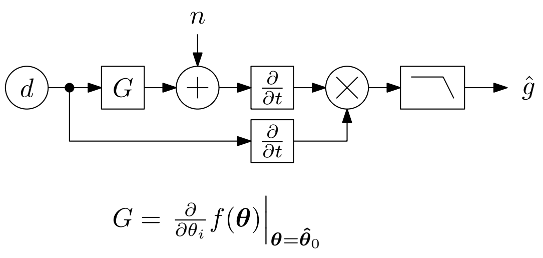
Taking the expectation of this expression:

$$\begin{aligned}
E\left\{\frac{\partial(dG+n)}{\partial t}\frac{\partial d}{\partial t}\right\} &= GE\left\{\left(\frac{\partial d}{\partial t}\right)^2\right\} + E\left\{\frac{\partial n}{\partial t}\right\}E\left\{\frac{\partial d}{\partial t}\right\} \\
&= GE\left\{\left(\frac{\partial d}{\partial t}\right)^2\right\}
\end{aligned}$$

In Figure 2.1 (b) a low-pass filter is used to perform a time-limited averaging, a short term expectation operation. [20] analyses the use of low-pass filters as expectation operators. However, [6] states that taking an average of the gradient brings no improvement for the convergence performance of SA algorithms — the summation operation inherently performs this averaging — and could be left out. The output of the system in Figure 2.1 (b) is G scaled by the variance of the dither source. p parallel copies of this system will compute an estimate of the p-dimensional gradient $\boldsymbol{G} = \boldsymbol{g}(\hat{\boldsymbol{\theta}}_0)$ of the loss function at the point $\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}_0$. Closing the loops with p integrators results in p identical scalar optimizers working in parallel, each trying to adjust its own weight.



(a) Continuous-time gradient descent optimization


n
d
G
$\frac{\partial}{\partial t}$
$\hat{g}$
$G = \frac{\partial}{\partial \theta_i} f(\boldsymbol{\theta}) \Big|_{\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}_0}$

(b) Open-loop single weight gradient estimation

Figure 2.1: Continuous-time gradient descent by simultaneous perturbation

Chapter 3

Analysis and design of smart antenna system

This chapter describes the analysis and simulation results of the mobile satellite TV smart antenna system. Figure 3.1 presents the various components which have been simulated and then implemented. The actual implementation and simulation consists of four variables but only two are shown for clarity. Figure 3.1 (a) shows the general splitting of the problem into two blocks: the *optimizer* circuit and the multidimensional *function* which needs to be optimized. The implementation was also split in this manner — the optimizer was implemented on a single printed circuit board which was then connected to the existing RF circuit. Figure 3.1 (b) shows a *test function* which, owing to its simplicity and predictability, is used to verify the performance of the optimizer in both simulation and implementation.

Figure 3.1 (c) shows the *antenna function*: the RF circuit configuration of the antenna array with microwave beamforming. The antennas, phase shifters and power combiner operate at 12.224–12.676 GHz. The low noise block (LNB) has a local oscillator (LO) frequency of 11.250 GHz to down convert the signal to 974–1426 MHz. The LNB is a necessary component required in all satellite TV applications for the low-loss transmission of the signal from the satellite antenna to the receiver through a long coaxial cable. The RF detector is a low-cost integrated circuit which obtains an averaged logarithmic measurement of the signal strength with an output (*video*) bandwidth of 3 MHz. This signal is the input to the optimizer which operates at frequencies up to 3 MHz. Note that the RF detector has eliminated a substantial amount of information by virtue of the averaging operation.

3.1 (d) shows the block diagram of the optimizer circuit performing a gradient descent operation. The gradient is estimated by correlating the function output with the *dither* sources which are added simultaneously to the function inputs, as explained in Section 2.5. A block diagram of the gradient estimator is shown in 3.1 (e). Comparing with Figure 2.1, the time derivatives are approximated as high-pass filters over the operating frequency range.

The optimization circuit was implemented using discrete components. The op-amp was the primary computational building block. The op-amp realizations of the various operations — taken from any introductory op-amp textbook — are presented in Figure 3.2. The design was simulated using the Agilent ADS circuit simulator, a simulator very similar to SPICE. The circuit was modeled as accurately as possible to minimize surprises in the implementation. Included were op-amp models based on data-sheet characteristics and all passive components such as resistors and capacitors. As a consequence, the optimizer simulation schematics (Appendix A) looks very similar to the implementation schematics (Appendix B).

The optimizer circuit is designed to operate on signals in the voltage range (-1, 1) V. To interface the optimizer circuit to an external function, such as the antenna function, simple shift and scale circuitry is needed on all inputs and outputs. In the case of the antenna function, the range (-1, 1) V maps to phase shifts in the range $(-\pi, \pi)$ radians. The dither noise added to each weight is fixed at about 100 mVpp which is equal to 5% of the total weight range of 2 V. It is assumed that the function output is appropriately scaled such that this small amount of dither does not see any function curvature. On the other hand, this dither was made large enough to produce a measurable effect on the function output.

The remainder of this chapter addresses in more detail specific components and issues and provides simulation results.

3.1 Test function

A very ideal target function, which we call the *test function*, is used in the simulation and the circuit implementation to test the optimizer (Figure 3.1 (b)). The two dimensional test function is any function which belongs to the family of functions which is defined by the following equation:

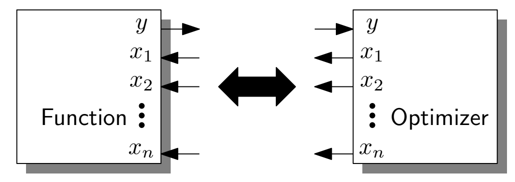
$$y = f(x_1, x_2) = -(x_1 - x_{1o})^2 - (x_2 - x_{2o})^2$$

x_{1o} and x_{2o} are constant function parameters. It would be very straightforward to extend this function to dimensions higher than two (e.g. $y = f(x_1, x_2, ..., x_n) = -(x_1 - x_{1o})^2 - (x_2 - x_{2o})^2 - ... - (x_n - x_{no})^2$) but order two was sufficient for both simulations and implementation. In the implementation only two (selectable) variables can be tested at the same time.

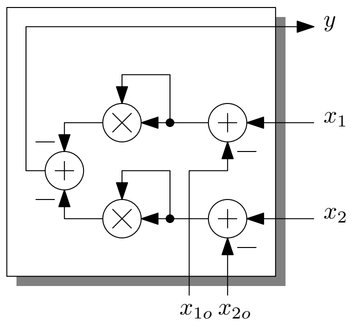
Figure 3.3 shows a plot of the test function and a derivative when $x_{1o} = x_2 = x_{2o} = 0$.

3.2 Antenna function

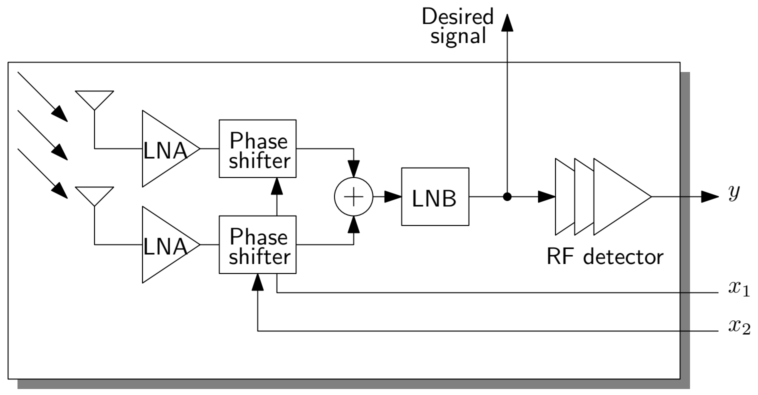
The real-world function which is the motivation of this work, called the *antenna function*, is any function from the family of functions which provides a received RF signal power measurement from the output of the antenna array, as a function of the phase shifter voltages (Figure 3.1 (c)). Many elements which make up this function are fixed, e.g. the antenna elements and their antenna pattern, the LNA characteristics, the antenna array configuration, but others are not, e.g. the array orientation with respect to the transmitter, the atmospheric conditions, the satellite transmitted power. We can group all variables which are not the function inputs x_i but which have an effect on the function output y and call them function parameters. Furthermore, we can assume the function parameters are constant during the short duration of time it takes to solve for



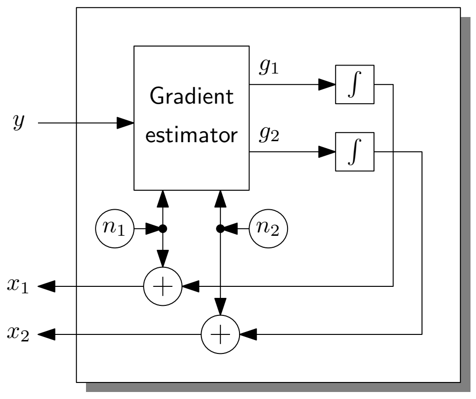
(a) Connections between optimizer and function



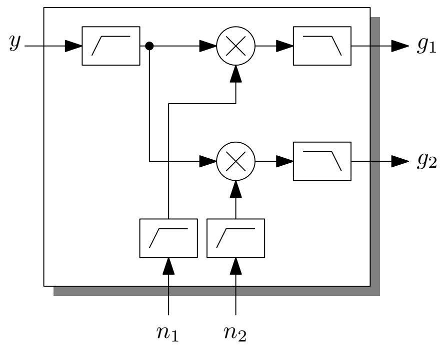
(b) Test function



(c) Antenna function

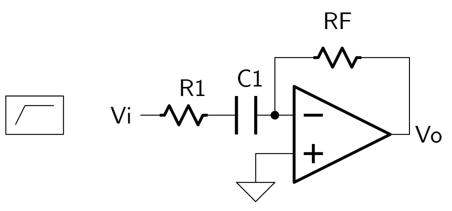


(d) Optimizer

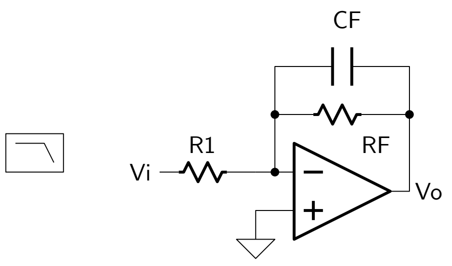


(e) Gradient estimator

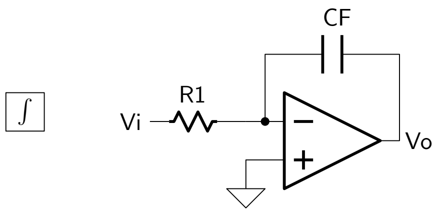
Figure 3.1: Smart antenna system components



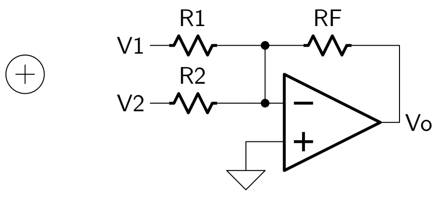
(a) High-pass filter



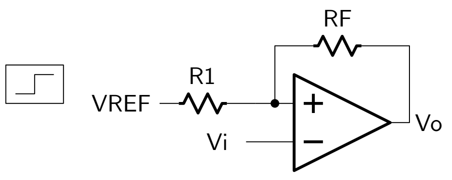
(b) Low-pass filter



(c) Integrator



(d) Adder



(e) Comparator with hysteresis

Figure 3.2: Circuit realizations

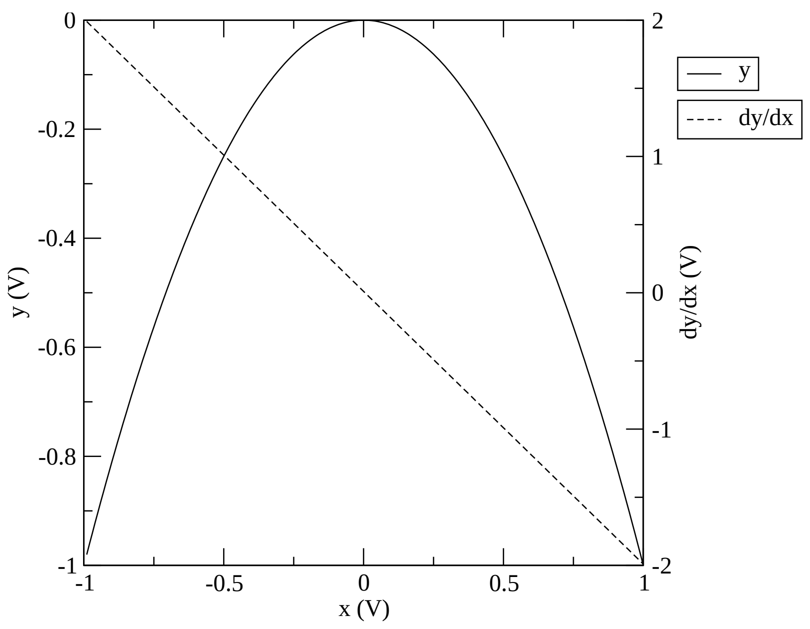


Figure 3.3: Test function and a derivative

the function maximum. In other words, we can work with the assumption that the received signal strength is only a function of the phase voltages. It is still important to see how the optimizer tracks a changing function, for example as the antenna orientation changes with respect to the satellite direction.

The real-world antenna function is very complex. A model is need which is as simple as possible while still retaining acceptable accuracy. The various components of the antenna function model are shown in Figure 3.4.

The phase shifters are assumed ideal phase shifters with phase shift equal to $\phi(x) = \pi x$, $x = [-1, 1]$ (Figure 3.4 (a) top). Everything up to the voltage signals at the outputs of the LNAs, which includes the electromagnetic environment, antenna orientation, array geometrical configuration, phase shifter phase errors, phase shifts due to mismatched path lengths, etc. is modeled by the block in Figure 3.4 (a):

$$v_i = e^{j\pi(x_i - x_{io})}$$

There is no restriction on x_{io} which are the sole function parameters which define the family of antenna functions. In reality, given various constraints only a limited subset of these functions could exist.

The power combiner is modeled as a 5-port power combiner built using 3 ideal 3-port Wilkinson power dividers (Figure 3.4 (b)). The S-parameters of the 3-port Wilkinson power divider (boxed inset) are given by ([21]):

$$\begin{bmatrix} V_1^- \\ V_2^- \\ V_3^- \end{bmatrix} = -\frac{j}{\sqrt{2}} \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^+ \\ V_2^+ \\ V_3^+ \end{bmatrix}$$

Therefore, assuming all matching circuits:

$$V_1^- = -\frac{j}{\sqrt{2}}\left(V_2^+ + V_3^+\right)$$

Repeating this three times (Figure 3.4 (b) middle):

$$\begin{aligned}
V_5^- &= -\frac{j}{\sqrt{2}}\left(V_1^+ + V_2^+\right)\\
V_6^- &= -\frac{j}{\sqrt{2}}\left(V_3^+ + V_4^+\right)\\
V_7^- &= -\frac{j}{\sqrt{2}}\left(V_5^+ + V_6^+\right)\\
V_7^- &= -\frac{1}{2}\left(V_1^+ + V_2^+ + V_3^+ + V_4^+\right)
\end{aligned}$$

The right-hand side of Figure 3.4 (b) shows graphically the phasor V_7^- as the composition of the 4 input phasors V_1^+ through V_4^+.

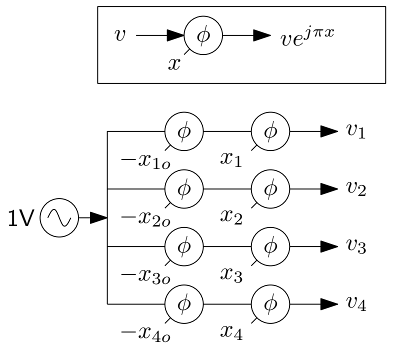
After the power combining, a low noise block (LNB) is used to step the frequency down and add further gain. The gain will just add a DC offset after logarithmic power detection so for simplicity a gain of 1 is assumed, or in other words the LNB is ignored. Then a single RF detector is used to measure the total received power, the AD8317 logarithmic detector from Analog Devices. Starting with the data-sheet of the RF detector and applying minor signal conditioning on the output, ignoring any DC offset at the output, the result will be typically $0.88\log(P_{in})\,V$ (Figure 3.4 (c)), which is the final output y of the function. Substituting all the individual components and normalizing to obtain a maximum value of zero, we obtain the following model for the antenna function $y = f(x_1, x_2, x_3, x_4)$:

$$\begin{aligned}
y &= 0.88\log\left(\frac{1}{4}\left|\frac{\sum v_i}{-2}\right|^2\right)\,V\\
y &= 0.88\log\left(\frac{1}{16}\left|\sum e^{j\pi(x_i - x_{io})}\right|^2\right)\,V
\end{aligned}$$

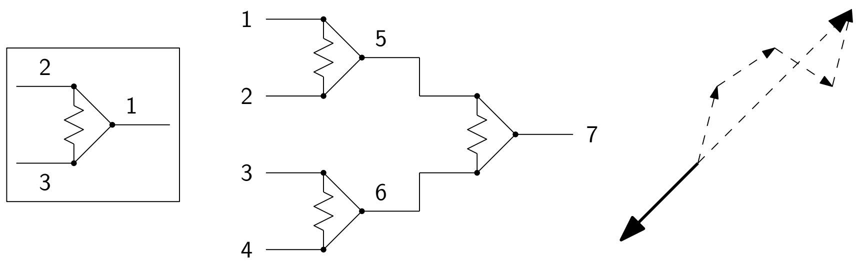
Figure 3.5 shows a plot of the antenna function and its derivative when $x_{1o} = x_{2o} = x_{3o} = x_{4o} = x_2 = x_3 = x_4 = 0$. One can see that it is fairly similar to the test function (Figure 3.3).

3.3 Analog noise sources

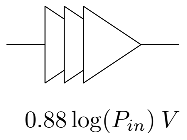
For this application we chose an efficient and simple method of generating the dither noise signals which is described by [14]. Using only a single 25-bit linear feedback shift register (LFSR)



(a) Environment, antenna, LNAs, phase shifters



(b) Power combiner



(c) RF detector

Figure 3.4: Antenna function components

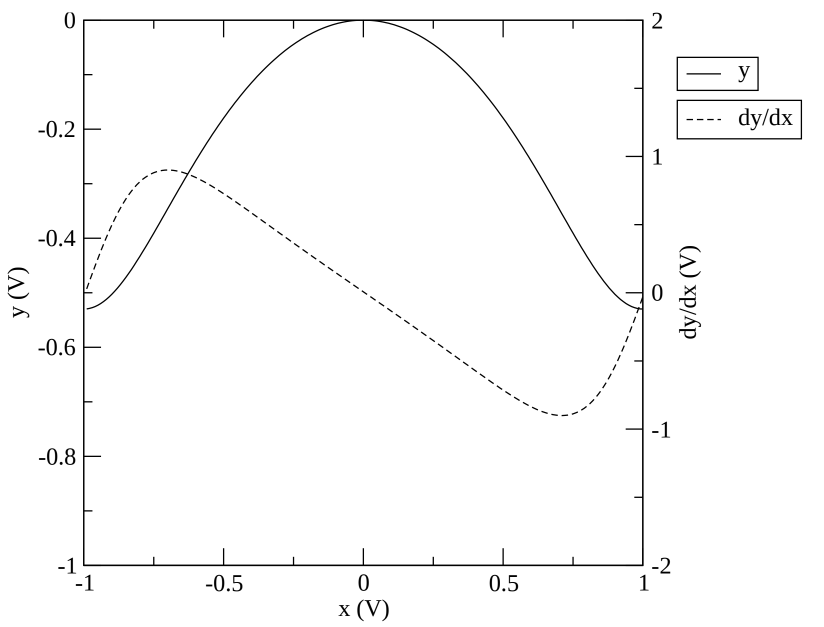


Figure 3.5: Antenna function and a derivative

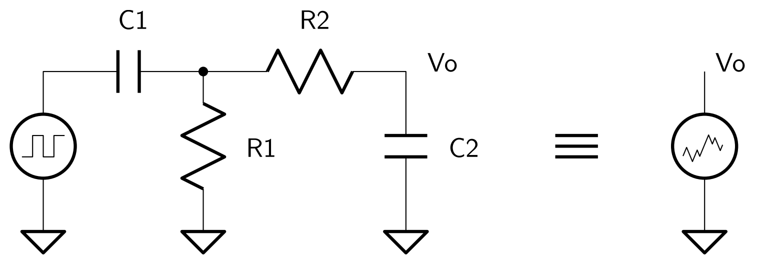
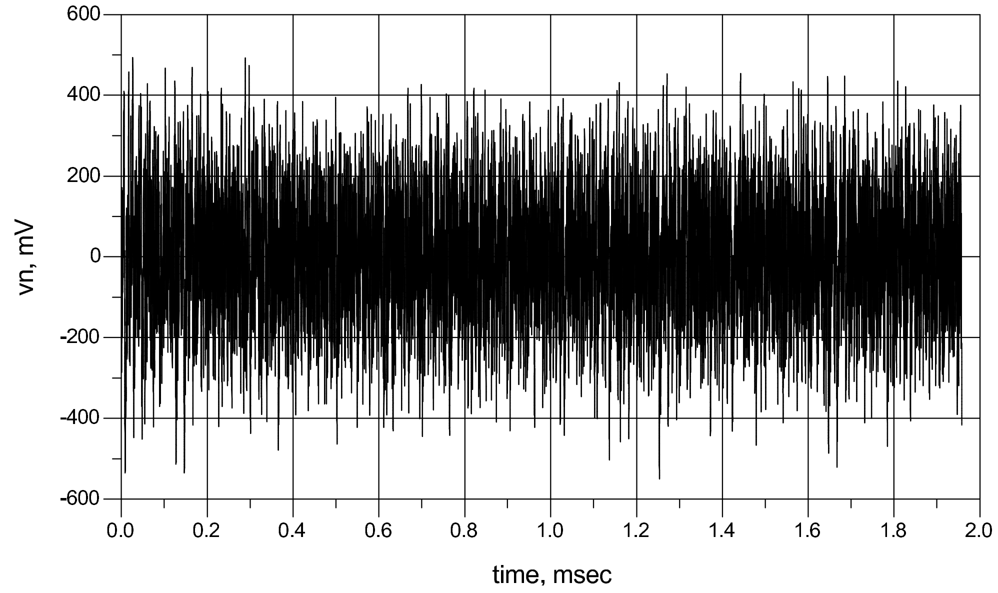
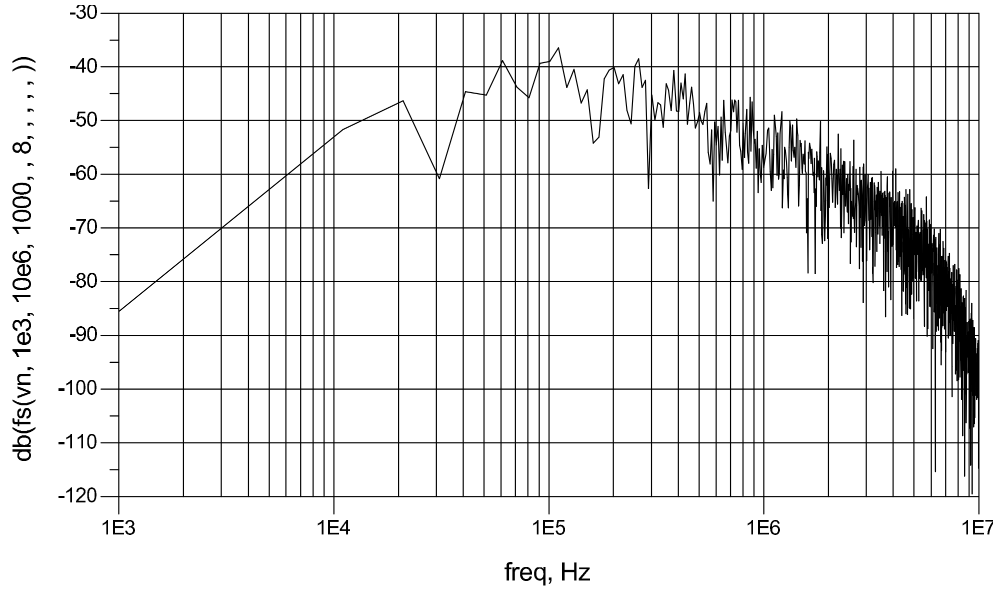


Figure 3.6: Noise source

32 mutually-uncorrelated digital bit sequences may be generated by carefully selecting the LFSR seed and XOR-combined taps. Each digital bit sequence is passed through a passive RC filter to produce a zero-mean analog noise signal as shown in Figure 3.6. Since the random bit sequence has a flat spectrum, the spectrum of the output analog noise signals resembles the transfer function of the filter. Figure 3.10 shows the frequency response of the filter chosen for this application and Figure 3.7 shows a transient simulation of the noise signal along with a Fourier transform showing the noise spectral density. Furthermore, the claims stated in [14] regarding the statistical properties of the noise sources (auto-correlation and cross-correlation) were verified. All 32 noise sources generate the same *master* bit sequence, but the bit sequences are delayed such that they are mutually uncorrelated for a period of time much longer than the expected convergence time of the algorithm, approximately $2^{25}/32$ bits. The master bit sequence repeats after 2^{25} bits, a period during which the auto-correlation is almost zero.



(a) Time domain



(b) Frequency domain

Figure 3.7: Noise signal characteristics

3.4 Gradient estimation non-idealities

Two important non-idealities exist during the gradient estimation, both of which may be lumped into the target function: (i) transient response of the target function and (ii) additional randomness present in the evaluation of the target function. The single variable gradient estimation circuit is shown in Figure 3.8 (see also Figure 2.1 and related discussion). The top sub-figure shows the ideal condition where one finds a gradient estimate $\hat{g}$ to the real scalar gradient g. The bottom sub-figure shows the more realistic non-ideal condition with both non-idealities included. These two non-idealities may be dealt with separately.

3.4.1 Target function transient response

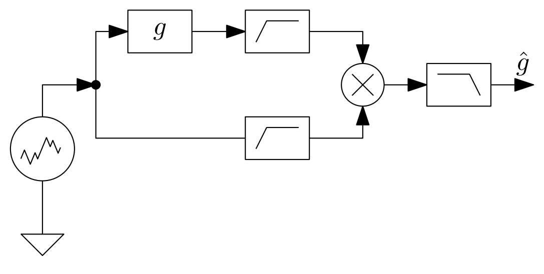
Consider replacing the noise source in Figure 3.8 (b) by a sinusoidal source of unit amplitude and setting the additive "noise" equal to zero. Up to the multiplier the circuit is linear and standard linear analyses apply. The two high-pass filters being identical, the multiplier sees two signals whose difference in magnitude and phase are governed completely by the transfer function $H(s)$. We assume $H(s)$ has a low-pass characteristic, a very common assumption which is valid in this particular application. Upon multiplying the two signals together, the output will see an attenuation due to the magnitude attenuation of $H(s)$ and another attenuation due to the phase mismatch. The magnitude attenuation scales the gradient estimate proportionately. This scaling does not constitute a serious problem because the gradient still has the same sign.

The phase mismatch attenuation can be calculated by finding the average DC value at the output of the multiplier as a function of phase mismatch (α):

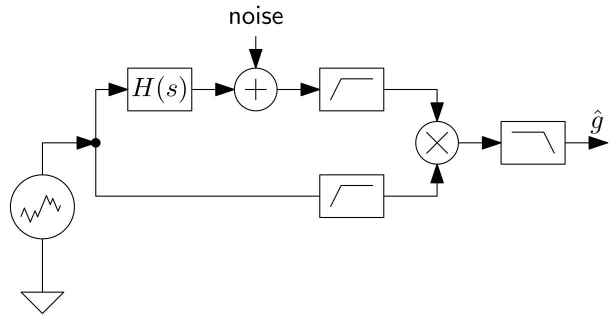
$$\begin{aligned}
\frac{1}{2\pi}\int_0^{2\pi}\cos(\omega t)\cos(\omega t+\alpha)dt &= \frac{1}{2\pi}\int_0^{2\pi}\cos(\omega t)^2\cos(\alpha)-\frac{1}{2}\sin(2\omega t)\sin(\alpha)dt \\
&= \frac{1}{2\pi}\pi\cos(\alpha)-0 \\
&= \frac{1}{2}\cos(\alpha)
\end{aligned}$$

We can normalize this and call it the correlation factor: $correlation\,factor=\cos(\alpha)$. A plot of the correlation factor vs. phase mismatch is shown in Figure 3.9. As expected, maximum correlation occurs when phase error is zero. At a phase error of 90° the correlation factor is zero. At phase error of 60° the correlation factor is 0.5. For the phase delay to be negligible, we find a correlation factor of 0.99 at a phase error of 8.1°. Beyond a phase mismatch of 90° the correlation factor becomes negative. This is a serious problem because the gradient estimate has the wrong sign which causes the algorithm solution to proceed in the wrong direction.

The phase mismatch can be circumvented in one of two ways: (i) compensate for $H(s)$ in the lower branch or (ii) design the optimizer to operate at a lower frequency where the phase mismatch is negligible (i.e. where the target function becomes ideal). The first solution has the advantage of being able to estimate the gradient at higher frequencies possibly resulting in a



(a) Gradient estimation with ideal function



(b) Gradient estimation with non-ideal function

Figure 3.8: Gradient estimation with non-idealities

faster algorithm. The second solution has numerous advantages including simpler design, less tweaking, and designing an optimizer which is suitable for applying to any target function which looks ideal up to a certain cut-off operating frequency. The second option was chosen in the present design.

3.4.2 Target function unwanted noise

The target function is evaluated in the presence of unwanted noise as in Figure 3.8. This noise consists of (i) actual injected dither noise in the other function variables and (ii) noise inherent in the measurement device. Both can be treated as unwanted noise in the context of estimating this single particular gradient component. Because this extra noise has zero mean and is uncorrelated with this particular injected dither noise, the algorithm will still converge but the convergence time may suffer due to gradient estimation errors.

The first type of unwanted noise — the dither noise injected by the other gradient estimators — cannot be filtered out because all dither signals share the same frequency domain.

The second type of unwanted noise is produced by different processes and most likely has a different probability distribution. For example, in the final system, performing satellite signal power measurements using the RF detector and minor signal conditioning reveals a noise signal with a flat spectrum and 90% peak-peak value of about 400 mV. In this particular application this completely unwanted noise has magnitude about four times greater than the injected dither noise! Fortunately, a large portion of this noise can be eliminated by choosing a narrow system

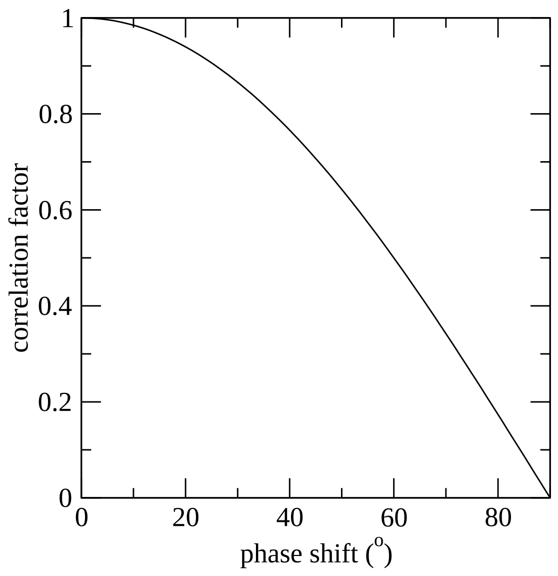


Figure 3.9: Correlation factor vs. phase shift

band of operation. The system bandwidth is determined by the combination of the noise filter and the high-pass filter, as will be described in more detail in Section 3.5.

3.5 Frequency domains

The analog gradient descent optimizer operates in multiple frequency regions. Figure 3.10 shows the magnitude of the frequency response of the most important components. The **test function** — this also applicable for the **antenna function** — appears in the top plot as a low-pass characteristic with -3 dB at about 3 MHz. The optimizer circuit works below this frequency to ensure that the function introduces no significant delay, as discussed in Section 3.4.

The **noise filter** exhibits a band-pass characteristic with center frequency at about 100 KHz, as shown in the second plot in Figure 3.10. The third plot shows the **high-pass filter** which implements the time derivative operation. It resembles the noise filter very closely since it is designed to emphasize only the desired injected dither noise. Both filters are in fact band-pass filters so that they block higher frequencies which are not of interest. The noise filter needs to have zero DC gain to ensure the dither signal has zero mean. The bandwidth and center frequency of these two filters define the *bandwidth of the gradient estimation operation*. As described in Section 3.4 the narrower the bandwidth the better the filtering of the out-of-band unwanted randomness in the target function.

By the nature of modulation, multiplication of two band-limited signals centered at frequency f_0 results in a new signal with one component having twice the bandwidth centered at DC and another component having twice the bandwidth centered at $2f_0$. This is exactly how the gradient estimate signal is generated. The gradient estimate needs to be low-pass filtered according to two competing criteria: (i) the cut-off frequency should be as large as possible to achieve a nimble gradient estimate and (ii) the cut-off frequency should be small enough to ensure that the gradient estimate does not have significant frequency components within the operating bandwidth of the gradient estimation operation. The last requirement can be restated from a different point of view as follows: the algorithm rate of descent should be much slower than the speed of estimating the gradient, to ensure that the weight updating does not affect the gradient estimation operation. If this criteria is not met, the positive feedback could cause instability. This filtering of the gradient estimate is achieved by the combination of the **low-pass filter** and the **integrator**, whose characteristics appearing in the bottom two plots of Figure 3.10 have been determined according to this criteria.

All transfer functions are first-order, resulting in gentle roll-off slopes of 20 dB/decade. As a future exercise, it might be instructive to analyze the improvements which may be obtained by choosing higher order filters to increase the slopes and make the bandwidth narrower.

3.6 Phase wrap-around

There is a problem specific to periodic functions such as the antenna function. If the optimal solution for a phase is x_0 there are in fact infinitely many solutions $x_0 + 2\pi i,\ i \in \mathbb{Z}$. This poses a problem for the optimizer which works in the range $(-\pi, \pi)$. Based purely on gradient descent, it is highly probable that the weight is optimized towards one of these extrema where it will become locked. For example, consider that the current phase is 0.9π and the ideal phase is 1.2π or equivalently -0.8π. The gradient descent operation will make the phase proceed towards π. At π the physical extent of the voltage is reached. The adopted solution is to make the phase must jump "instantaneously" to the equivalent phase of $-\pi$ and continue from there. This is shown graphically in Figure 3.11. A circuit which does precisely this jump has been modeled and implemented. It works by quickly transferring a precise charge to the integrator. The schematics may be found in Appendix A (simulation) and Appendix B (implementation). The jump must be fast enough to be invisible to the gradient descent algorithm. A symmetric situation occurs at the negative extreme, at $-\pi$. In order to avoid an eternal sequence of jumps from one extreme to the other, a hysteresis of 10% is provided. In other words, the positive jump occurs from 1.1π to -0.9π and the negative jump occurs from -1.1π to 0.9π. The hysteresis is also shown in Figure 3.11.

3.7 Simulation results

Transient simulations were performed using Agilent ADS circuit simulator. Three main simulation results are presented: (i) the two-variable test function without any extra added noise, (ii) the test function with additional noise, and (iii) the four variable antenna function with one variable held constant. The results are presented in Figures 3.12, 3.13, and 3.14 respectively. Each result displays the function output and its time derivative. For each variable three plots are shown: (i) the variable value (top), (ii) the instantaneous gradient estimate measured at the output of the multiplier (bottom), and (iii) the averaged gradient estimate produced after the low-pass filter (middle). Several sign inversions exist in the circuit for implementation reasons. One such inversion applies to the instantaneous gradient (bottom plot).

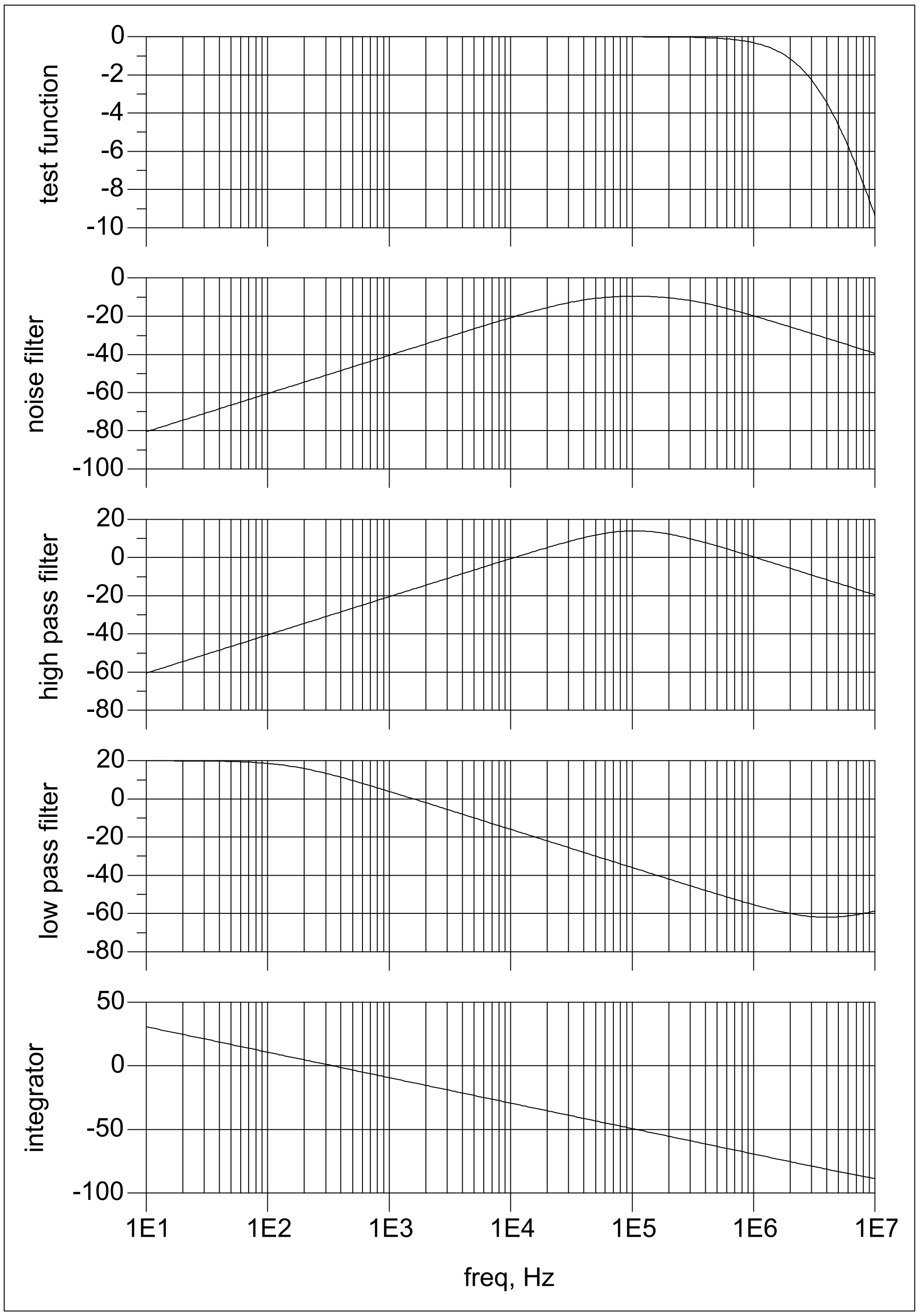


Figure 3.10: Frequency bands in the optimizer circuit (magnitude in dB)

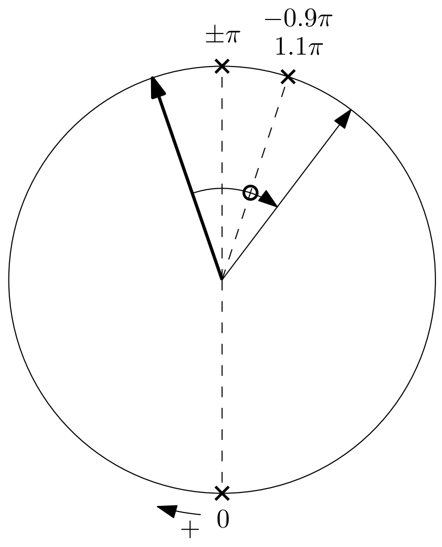


Figure 3.11: Phase wrap-around in the positive direction

Figure 3.12 shows the transient response during a step change between two test functions. Recall the test function is $y = f(x_1, x_2) = -(x_1 - x_{1o})^2 - (x_2 - x_{2o})^2$, parametrized by x_{10} and x_{20}. Two test functions are created: $f_1(x_1, x_2) = -(x_1 + 0.6)^2 - (x_2 - 0.4)^2$ and $f_2(x_1, x_2) = -(x_1 - 0.3)^2 - (x_2 + 0.5)^2$. At time zero an instantaneous step occurs from $f_1(.)$ to $f_2(.)$, starting from an initial state where the algorithm is fully converged on the maximum of $f_1(.)$. The results show how the algorithm converges on the maximum of $f_2(.)$. After about 15 ms the function output has returned to within 90% of its maximum. The two gradient components drive the variables x_1 and x_2 simultaneously to the new optimum values. One can see the function derivative diminishing in magnitude and consequently the gradient estimates approaching zero as the maximum is approached. The instantaneous gradient estimate looks very noisy although for the most part maintaining the proper sign and the averaged gradient estimate looks very clean.

In Figure 3.13 the same scenario is repeated but approximately 0.2 Vpp of wide-band noise is added to the function output. The function output and the time derivative looks much noisier than before, even when approaching the maximum. The instantaneous gradient estimate also looks very noisy often the sign being erroneous. However the averaged gradient estimate looks similar to the previous noise-free experiment. Most importantly, the convergence time is virtually unaffected by the extra noise.

To estimate the behavior of the smart antenna system a transient simulation is performed using a step change between two antenna functions (Figure 3.14). Recall the antenna function parametrized by x_{io}:

$$y = f(x_1, x_2, x_3, x_4) = 0.88 \log\left(\frac{1}{16}\left|\sum e^{j\pi(x_i - x_{io})}\right|^2\right)$$

One weight is kept constant at $x_4 = x_{40} = 0$ forming a three variable function. Two functions $f_1(.)$ and $f_2(.)$ are created by choosing two sets of parameters $\{x_{i0}\}_1$ and $\{x_{i0}\}_2$. At time zero an instantaneous step occurs from $f_1(.)$ to $f_2(.)$, starting from an initial state where the algorithm is fully converged on the maximum of $f_1(.)$. The results show how the algorithm converges on the maximum of $f_2(.)$. The 90 % convergence time occurs a little bit beyond the recorded simulation time and is estimated at about 30 ms. This simulation shows how the convergence time can vary based on the function being optimized.

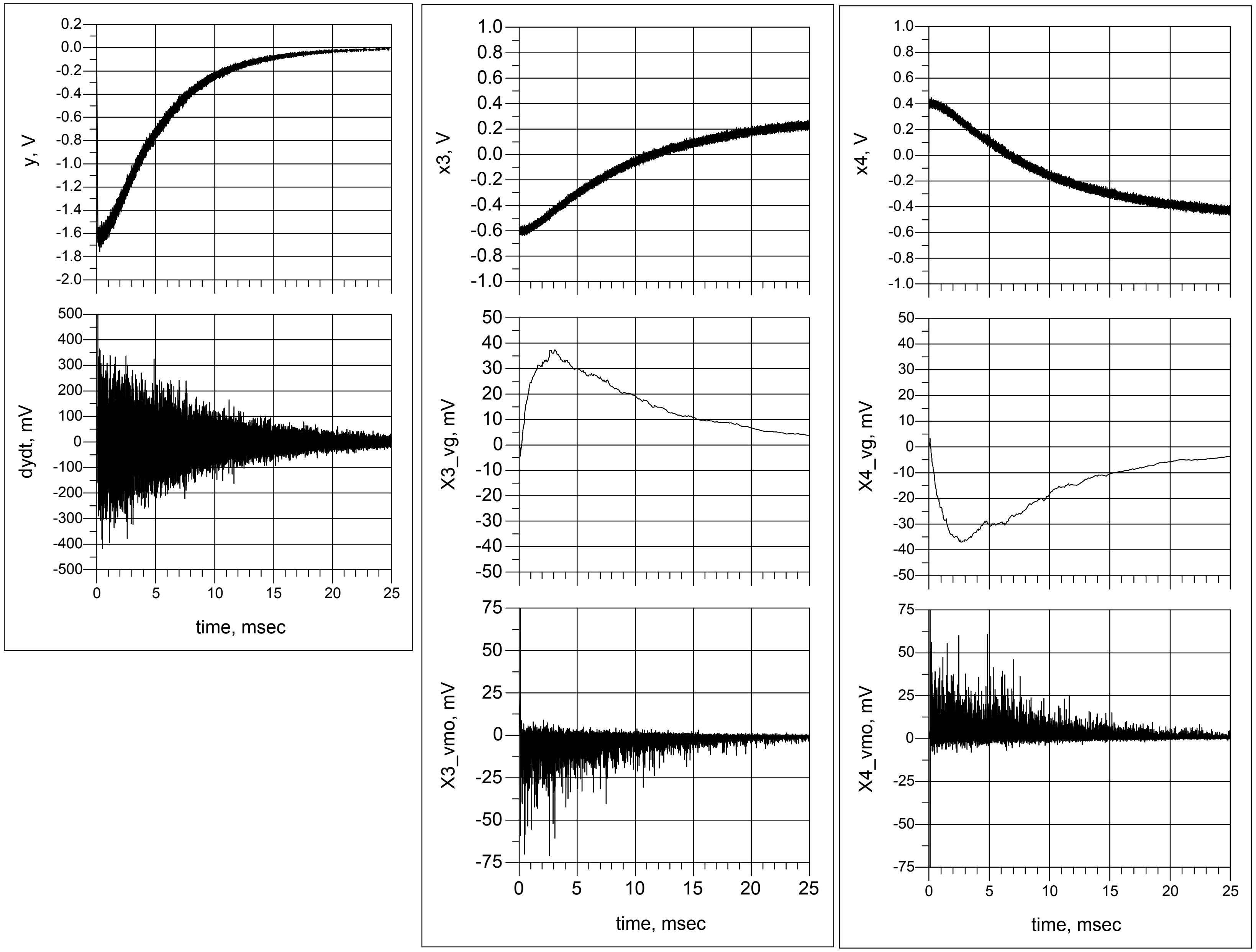


Figure 3.12: Simulation of test function transient response without measurement noise

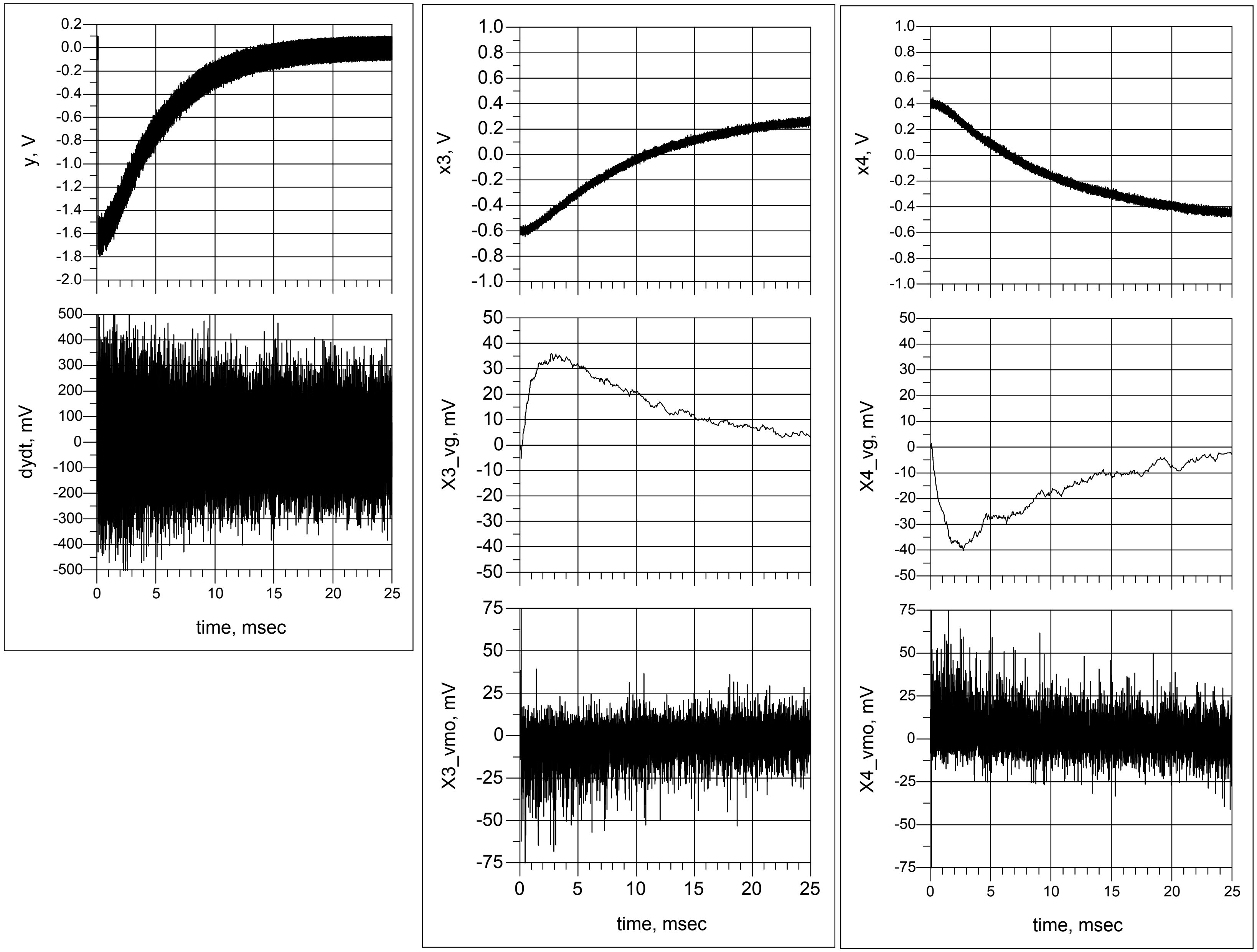


Figure 3.13: Simulation of test function transient response with measurement noise

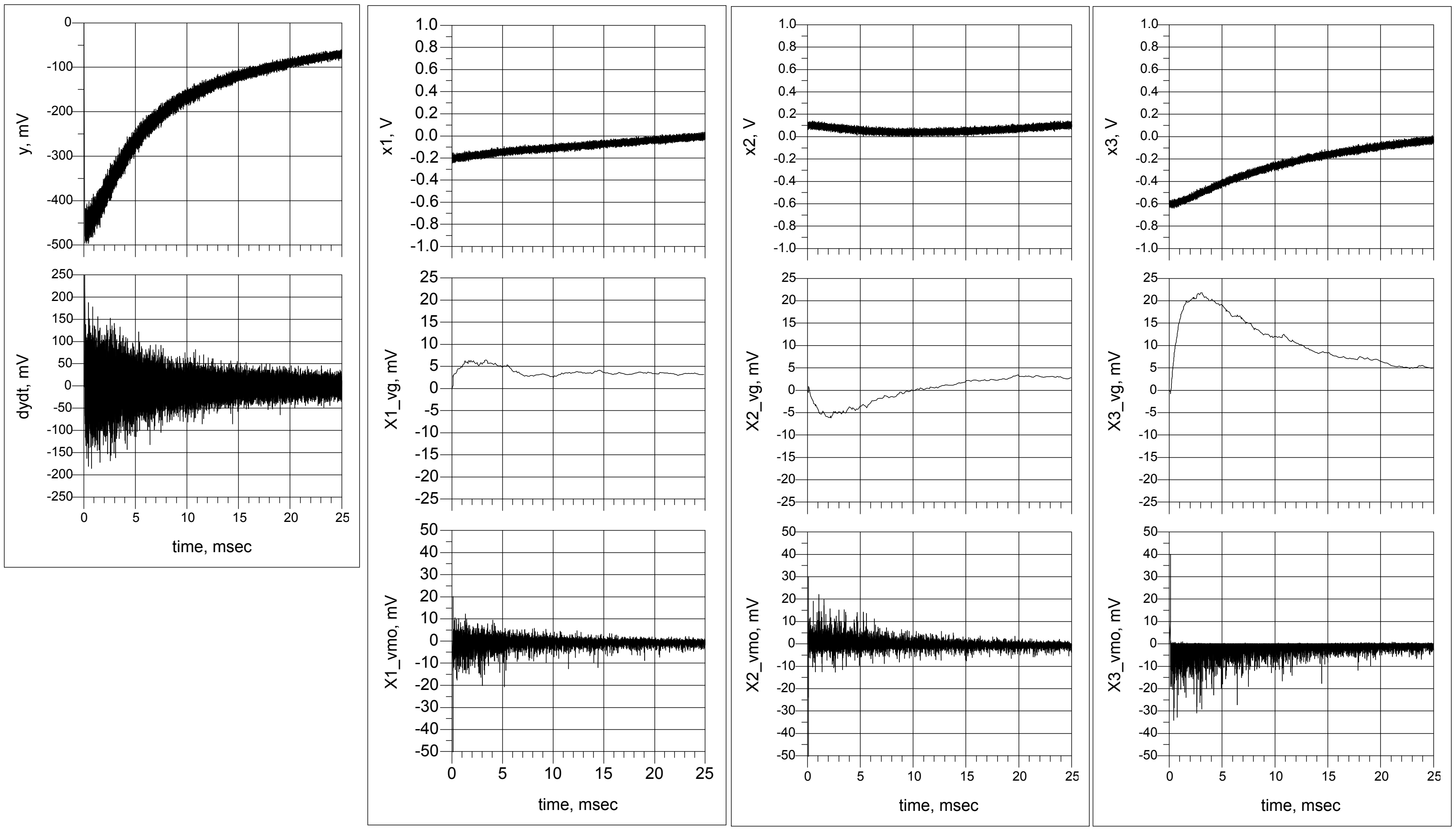


Figure 3.14: Simulation of antenna function transient response

Chapter 4

Implementation and results

Based on the simulation results from the previous section a four variable optimizer circuit board was designed built and tested. This section presents a description of the optimizer board functionality, a description of the experimental procedures, and the obtained results.

4.1 Implementation

Figure 4.1 shows a block diagram of the optimizer board. The complete schematic may be found in Appendix B. Only two variables are shown for clarity. The dashed box outlines a one-variable optimizer; this is the main unit which is duplicated p times to produce a p-variable optimizer (for this implementation $p = 4$).

The board has a stepper motor interface, a digital interface, as well as an analog function interface.

The digital interface consists of a parallel interface and a serial peripheral interface (SPI). The parallel interface connects to a complex programmable logic device (CPLD), which provides control of the test function, analog switches, stepper motor, and many other simple on-off type functions on the board. The SPI interface provides direct access to the digital to analog converters (DACs) and analog to digital converters (ADCs). The CPLD also generates the digital pseudo-random noise sources which become the analog dither noise sources. The CPLD source code is written in Verilog.

The block labeled "test function" implements the test function described in Section 3.1: $y = f(x_1, x_2) = -(x_1 - x_{1o})^2 - (x_2 - x_{2o})^2$. Only a two dimensional test function is implemented, but using switches different variables may be routed to the test function. The function may be configured by modifying the parameters x_{io} which is accomplished by a set of DACs controlled by the CPLD. One switch selects between the test function and antenna function as the target function. The function output can be sampled using an ADC and the data transferred to a computer.

A power supply provides both positive and negative supplies used to operate the circuit. A separate block labeled "test voltages" provides hand-tunable precision reference voltages for adjusting various circuit parameters. A stepper motor driver adds one-axis rotation capability to the antenna array.

The block diagram shows two one-variable optimizer units, one being outlined by a box. The digital noise from the CPLD is filtered to produce an analog noise source. A high-pass filter is then applied to the noise implementing the time-derivative operation. The result is multiplied by the identically filtered output of the target function. The next step is low-pass filtering which completes the gradient estimation. The gradient estimate is passed through an integrator to produce the gradient descent operation by which the weights migrate to their optimum values. The two wrap-around circuits work together to implement the wrap-around operation described in Section 3.6. One circuit performs a rapid transition $1.1V \rightarrow -0.9V$ and another performs the symmetrical transition $-1.1V \rightarrow 0.9V$. The loop can be broken with the switch and the weight controlled by software using a DAC, an essential function for performing various tests and trying software-based algorithms. The weight can be sampled using an ADC and the data transferred to a computer. The last operation is a weight scaling and shifting to meet the requirements of the external target function, in this case the phase-voltage characteristics of the phase shifter.

A micro-processor board connects to the digital interface. The micro-processor board consists of 32-bit Freescale Coldfire embedded system. This was a purchased development board. The purpose of this board is to provide higher-level time-critical functionality to the system, such as data collection from the ADCs. The source code is written in C and uses the small operating system WhatOS.

A computer connects to the micro-processor board using a serial port. The software on the computer accesses the low-level simple functional building blocks provided by the CPLD code and the micro-processor code. The computer code is written in the high-level scripting language Python.

A labeled photograph showing the final assembled system is shown in Figure 4.2.

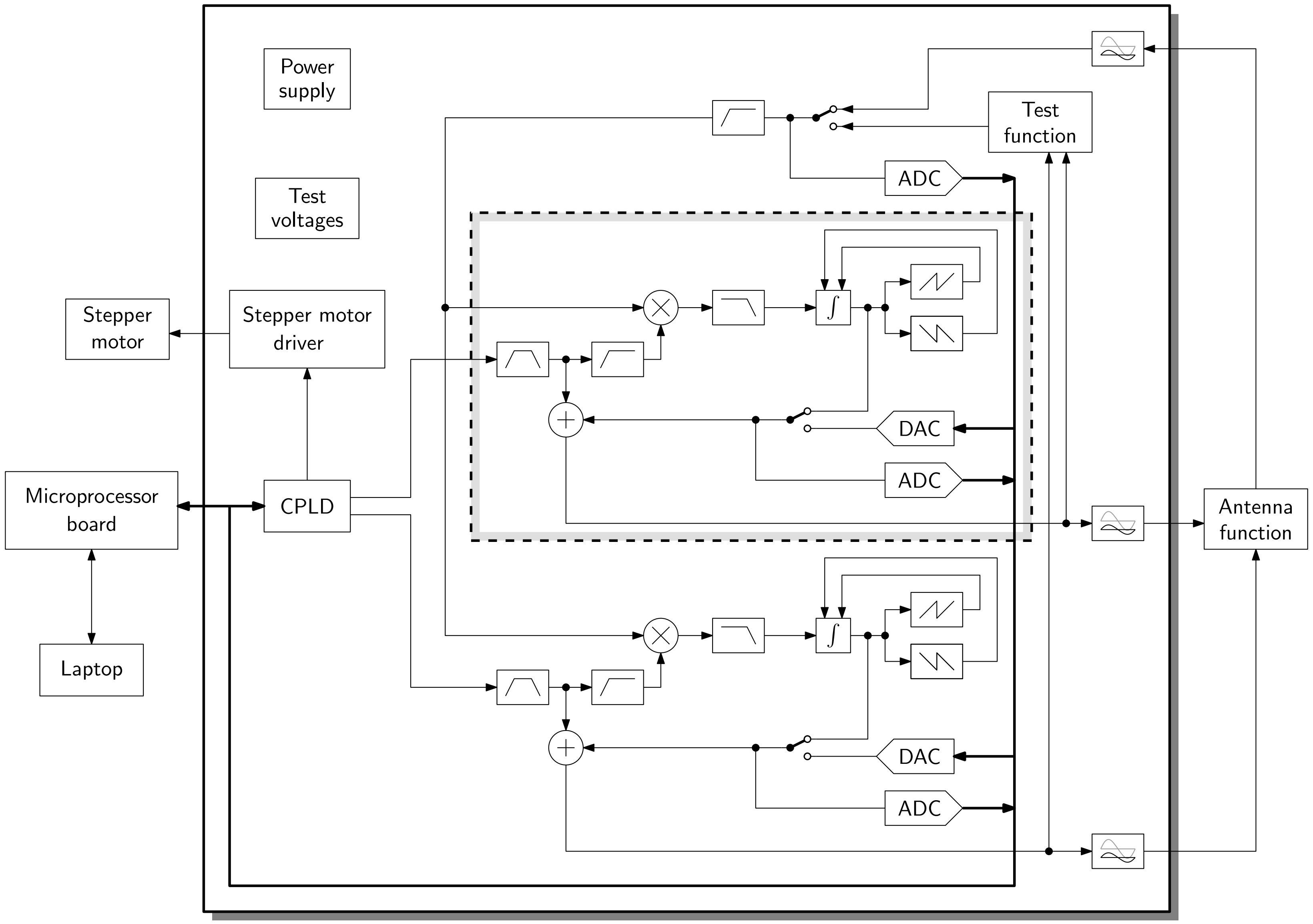


Figure 4.1: Optimizer board block diagram

Table 4.1: Individual antennas contributions

Disabled antenna	Received power (V)	Δ (V)
none	1.056	0
1	0.946	-0.110
2	0.923	-0.132
3	0.874	-0.181
4	0.881	-0.175

4.2 Verification measurements

A set of measurements were performed to verify the operation of the circuit. Here we will present a few of these. The RF detector step response and noise characteristics are measured by applying a step to one of the phases and capturing the RF detector output. The oscilloscope capture is presented in Figure 4.3. While the antenna is receiving a signal, a step is applied to one of the weights (top trace) resulting in a simultaneous step in the corresponding phase, which in turn causes the received signal power to change. The bottom trace shows the RF detector signal reflecting the change in received power. It shows that for a 2 μs step the delay is virtually unperceivable. Furthermore a noise of about 400 mVpp is perpetually present in the RF detector measurement.

The generated analog dither noise signals were measured; one of these is shown in the oscilloscope capture in Figure 4.4. The amplitude is 1 Vpp as predicted by the simulation (Figure 3.7). (This signal is multiplied by a factor of 0.1 before being added to the weight, resulting in a dither amplitude of about 100 mVpp). The noise signals were also analyzed spectrally and statistically and the simulation predictions were confirmed.

The contributions provided by each of the four antennas is characterized in the following way. The algorithm is performed to optimize the received satellite signal power level. Then each of the four antennas are disabled one at a time, by turning the power off to the corresponding LNA. The results are presented in Table 4.1. Each row corresponds to one disabled antenna; the exception is the first row where no antennas are disabled. The second column is the RF detector measurement. The third column shows the drop in signal power compared with the case where no antennas are disabled. Ideally all the numbers should be the same if we assume that all RF paths — which includes antennas, LNAs and power combiner — are identical. In practice it is apparent that this is not so. However, the obtained measurements show that all antennas contribute to the received signal strength and the contributions are similar enough to indicate that the system is operating correctly.

4.3 Experimental results

There are two categories of tests which were performed: *final value tests* and *transient tests*. Tests were performed using the two target functions: the test function and the antenna function. The antenna tests were performed outside using a satellite signal. The response to different initial

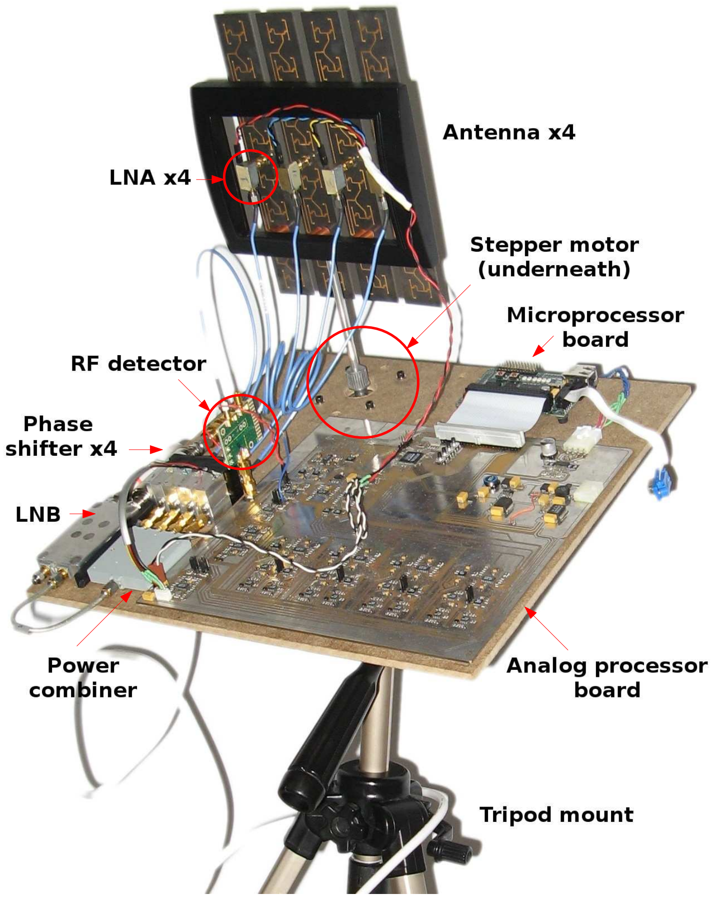


Figure 4.2: Complete system photograph

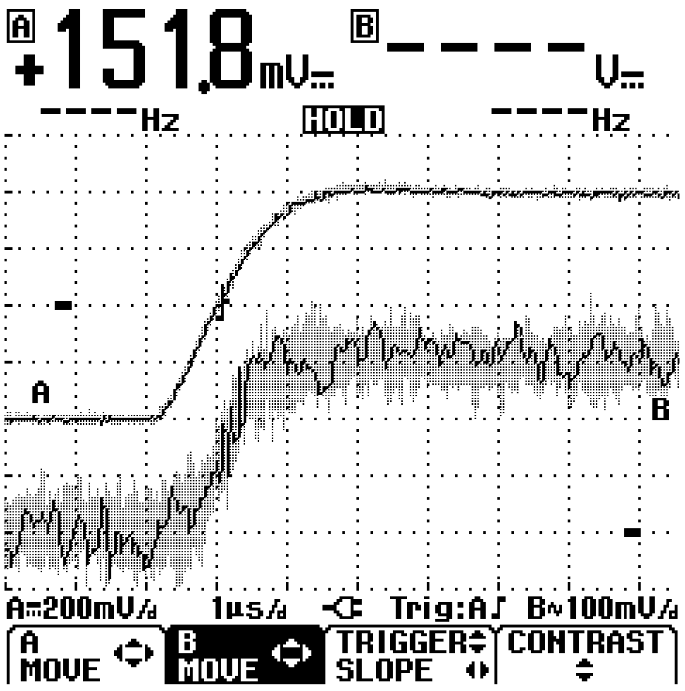


Figure 4.3: RF detector step response and inherent noise

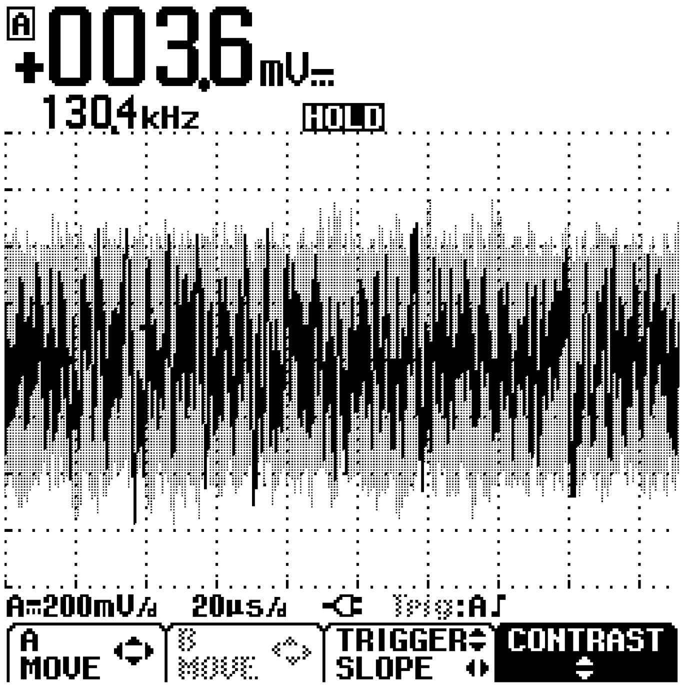


Figure 4.4: Generated dither noise

states, as well as the response to changing target function were measured. Some of the tests performed permit a meaningful comparison with the simulations.

The analog algorithm was compared with a similar digital algorithm implemented in software on the computer. The software algorithm is the finite difference stochastic approximation (FDSA) described in Section 2.4.1. The algorithm was hand-tuned for achieving the best accuracy and an acceptable convergence time. Given the difference in implementation comparison on the basis of speed would be meaningless. Therefore the gradient estimation was made very slow but accurate. Typical convergence time is about 30 seconds. The weight wrap-around technique is also implemented in this software algorithm.

Keeping with the convention established thus far, all results will show the weights and function output *voltages*. All signals have a valid range of (-1, 1) V. One should keep in mind that in the case of the antenna function the phase angle is approximately equal to πx_i and the function output y is related to the received signal power by the approximate relation $y = 0.88 \log(P_{in})\, V$ or equivalently 88 mV per dB. These approximate relationships are based on typical measurements, but were not accurately characterized or calibrated, an important future task.

4.3.1 Received signal strength vs pointing direction

This experiment is performed outside with a satellite signal. The goal is to measure the signal strength received by the smart antenna as a function of mechanical pointing deviation from ideal. Mechanical motion is only possible in one axis; the other axis is manually optimized once at the beginning of the set of experiments. The position scanning is performed from the left angle (negative) through perpendicular (zero) through right angle (positive).

An important reference point is the maximum achievable signal level. Given that the antenna elements have maximum gain in the direction perpendicular to the antenna plane, the maximum achievable signal level from the smart antenna should occur when the satellite direction is perpendicularly incident on the antenna array plane, while the phases are such that the individual signals combine in phase. This was achieved by the following calibration procedure. Starting with all phases fixed at zero, the antenna array is pointed optimally towards the satellite. The digital algorithm is then performed and the solution is frozen. Again the antenna array is pointed optimally. This is repeated until there is no more observed improvement in signal strength. The resulting mechanical orientation of the antenna will be called the *perpendicular position*, and the resulting phase shifter phases will be called the *perpendicular phases*.

The first test measures the received signal level as a function of orientation when the algorithm is off. The scenario is shown in Figure 4.5. The phases are fixed at the perpendicular phases and the antenna array is moved to the left and right of the perpendicular position. The result is shown in Figure 4.6 (top). As expected the signal is strongest at the perpendicular position and drops off on either side. This test basically gives the antenna array pattern with phases fixed. This test is also used to identify a reasonable range of mechanical deviation for the remaining tests. Beyond 7° to the right of perpendicular the function begins to turn upward; therefore the range of mechanical deviation for the remaining tests was chosen as $(-5.4°, 5.4°)$. The experiment was repeated over this limited range producing the results in Figure 4.6 (bottom). This figure is to be used as a reference for comparing the improvements obtained with various adaptive algorithms.

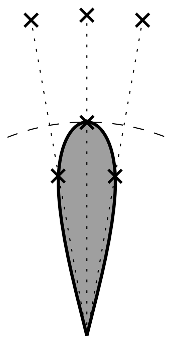
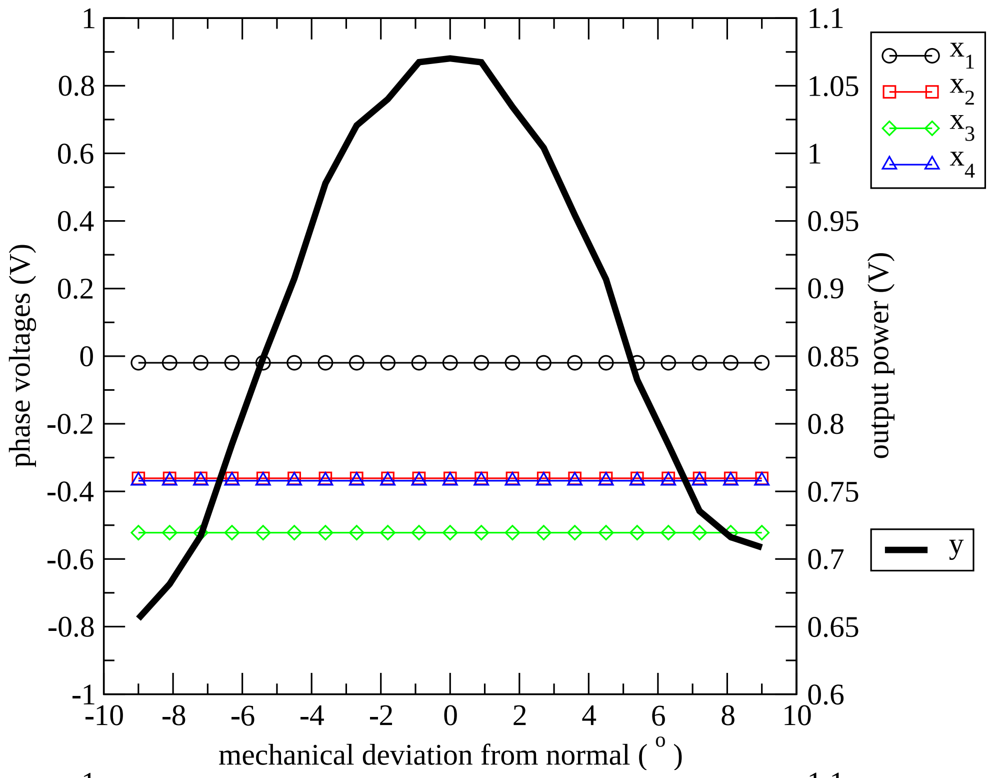


Figure 4.5: Signal strength vs. pointing direction with algorithm off

In the next experiment the analog optimizer is turned on for all phases except the phase corresponding to x_1. The test scenario is shown in Figure 4.7. The analog optimizer is initialized only once at the beginning of the experiment, at the perpendicular orientation, after which it is kept operating for the remaining of the experiment. Two separate experiments are performed about ten minutes apart to verify the repeatability of the experiment. The results are presented in Figure 4.8. The similarity of the two graphs confirms the repeatability of the experiment. The received power level never drops below 1.025 V, a big improvement over the result obtained with the fixed phases (Figure 4.6). The phase corresponding to x_4 experiences positive wrap-around, confirming the operation of the wrap-around circuit.

In the next experiment the digital optimizer is turned on for all phases except the phase corresponding to x_1. The digital optimizer is initialized only once at the beginning of the experiment, at the perpendicular orientation. After waiting the algorithm to converge, the algorithm is stopped and the antenna is positioned at the most negative angle. The algorithm is then allowed to continue until it again converges, after which it is stopped and the resulting power is recorded. This is repeated for every angle. The results are presented in Figure 4.9. Once again two separate experiments are performed approximately 10 minutes apart to see the repeatability of the experiment; the similarity of the results confirms this repeatability. It seems that the digital optimizer finds a different set of solutions than the analog optimizer, but the effectiveness of both solutions is very similar — the output power does not drop below 1.025 V. The digital algorithm being the benchmark, comparing Figures 4.8 and 4.9 shows that the analog optimizer performs almost identically as well as the digital optimizer.

Comparing the solutions found by the analog optimizer and the digital optimizer, as well as other field tests, it became apparent that for the same antenna function the optimizer might converge on different solutions depending on initial state. This scenario is described graphically in Figure 4.10. In order to verify that this is the case the previous two tests were repeated, but this time the analog and digital optimizers were reinitialized with a random state five times at each antenna position. The result for the analog optimizer is shown in Figure 4.11. At every position there appear to be multiple steady states, some resulting in convergence to a suboptimal function output. The output power and the phases show some clustering. The result for the digital optimizer is shown in Figure 4.12. Again there are multiple steady states, some resulting in suboptimal function output. In this case the solutions seem to be more closely clustered together, but there are some blatant outliers such as the poor solution at the position of 2.7°.



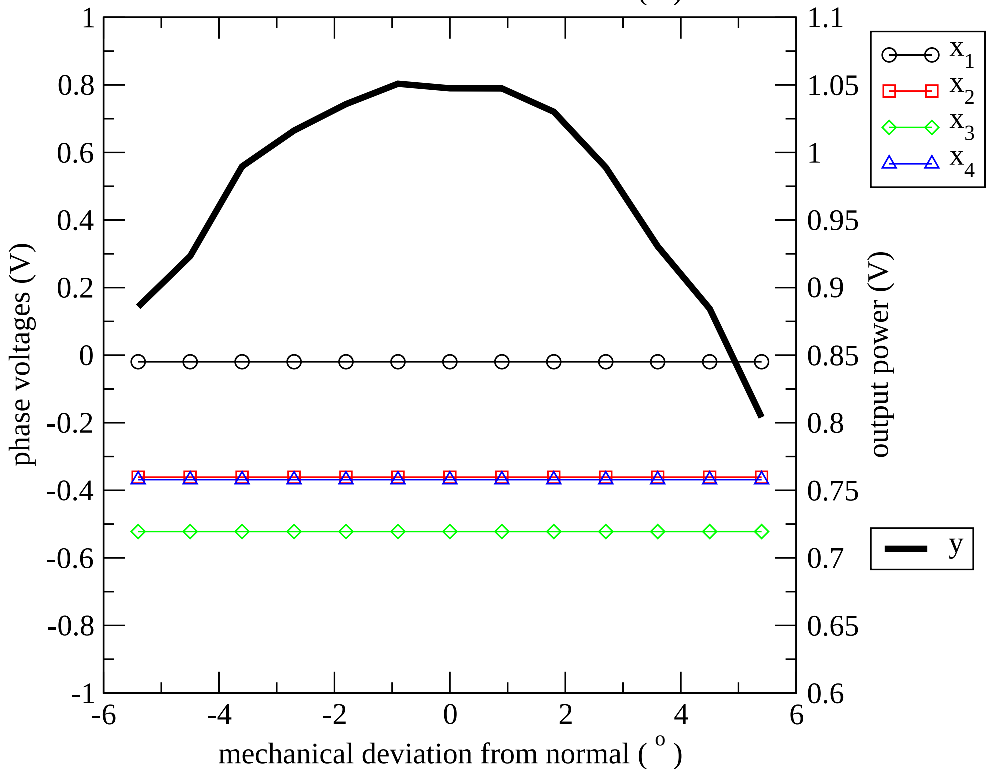


Figure 4.6: Power vs direction with fixed phases

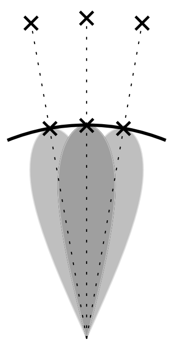
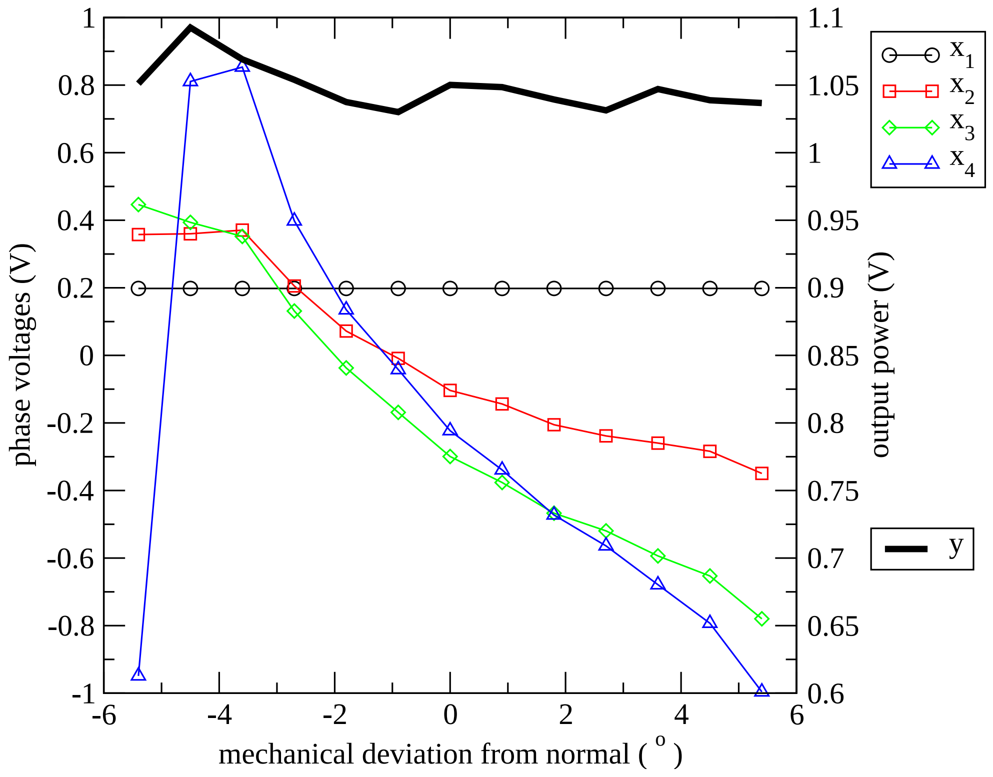
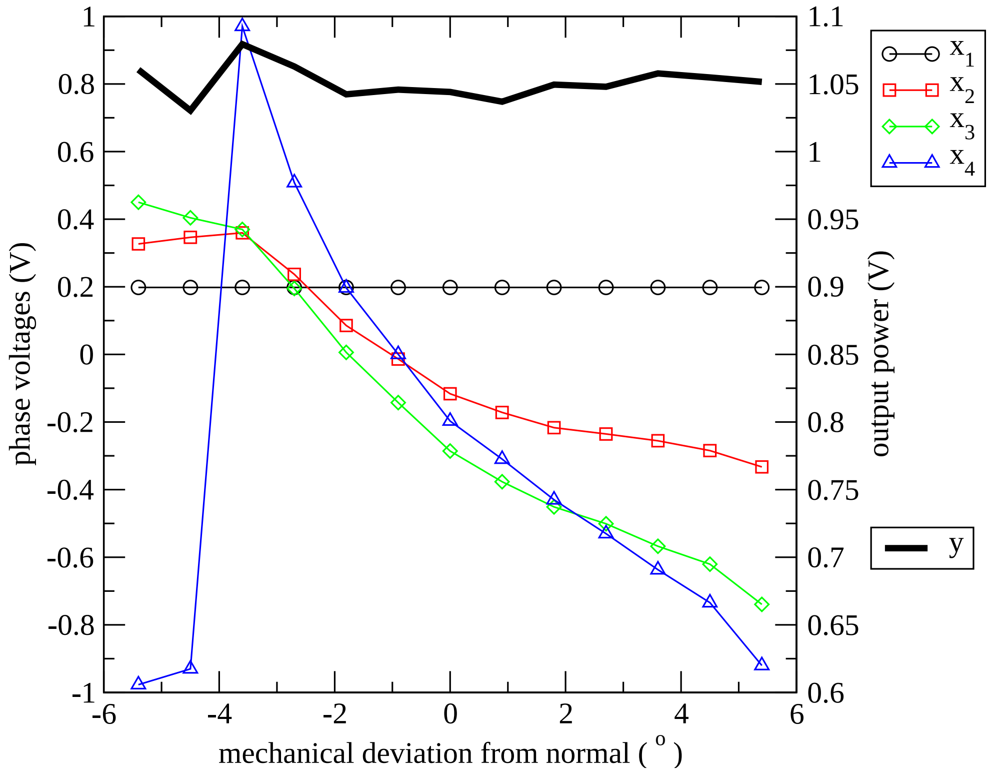


Figure 4.7: Signal strength vs. pointing direction with algorithm on

The digital algorithm provides a much more “organized” graph in this case and the reason for this difference is a good subject for future investigation. What seems to be indisputable is the fact that the analog function has multiple local maxima which were not modeled by the simple analog function model described in Section 3.2. One possible explanation may be that the antenna array sometimes creates a beam towards neighboring satellites, depending on initial state. This possibility should be modeled and investigated further. Another explanation, which does not exclude the previous one, could be the phase shifter phase-gain relationship. It was discovered late that the phase shifter has a non-ideality which causes the gain to be a function of control voltage (see Figure 1.6). Whereas the phase shifter was modeled as $y = e^{j\pi x}$ the actual model looks like $y = f(x)e^{jg(x)}$. Function $g(x)$ is a monotonically increasing function and besides an effect on the convergence rate the final value should not be affected. On the other hand function $f(x)$ can cause multiple solutions since a non-aligned phase may be preferred over the aligned phase due to much greater gain. An improved phase shifter model should definitely be included in future simulations.

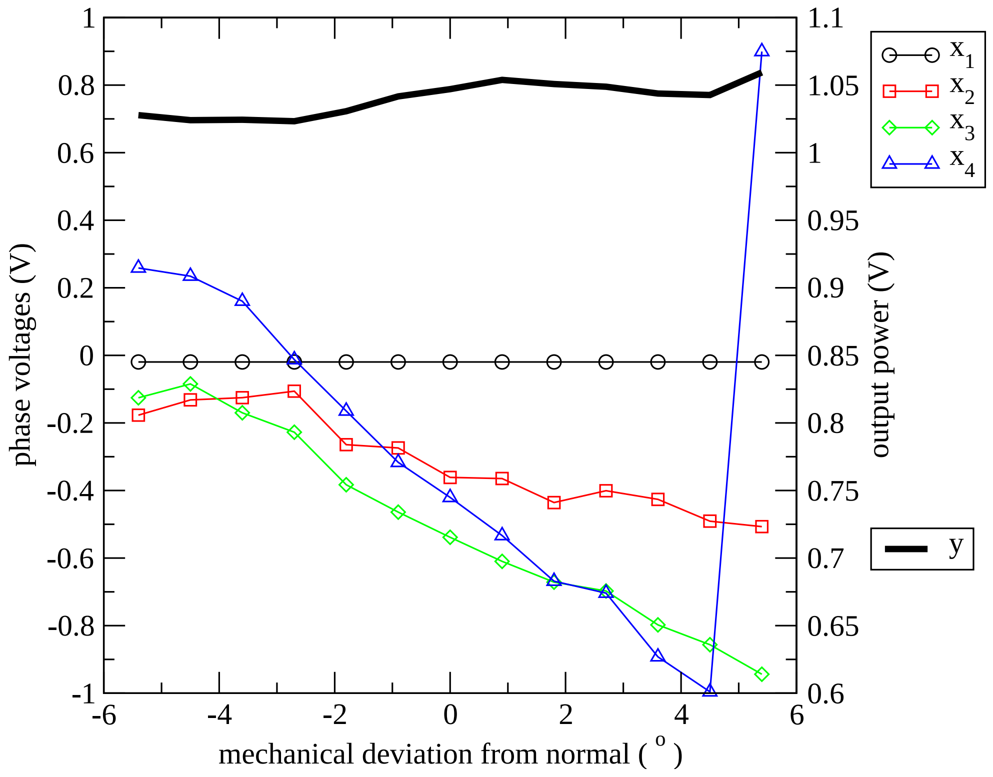


(a) First test

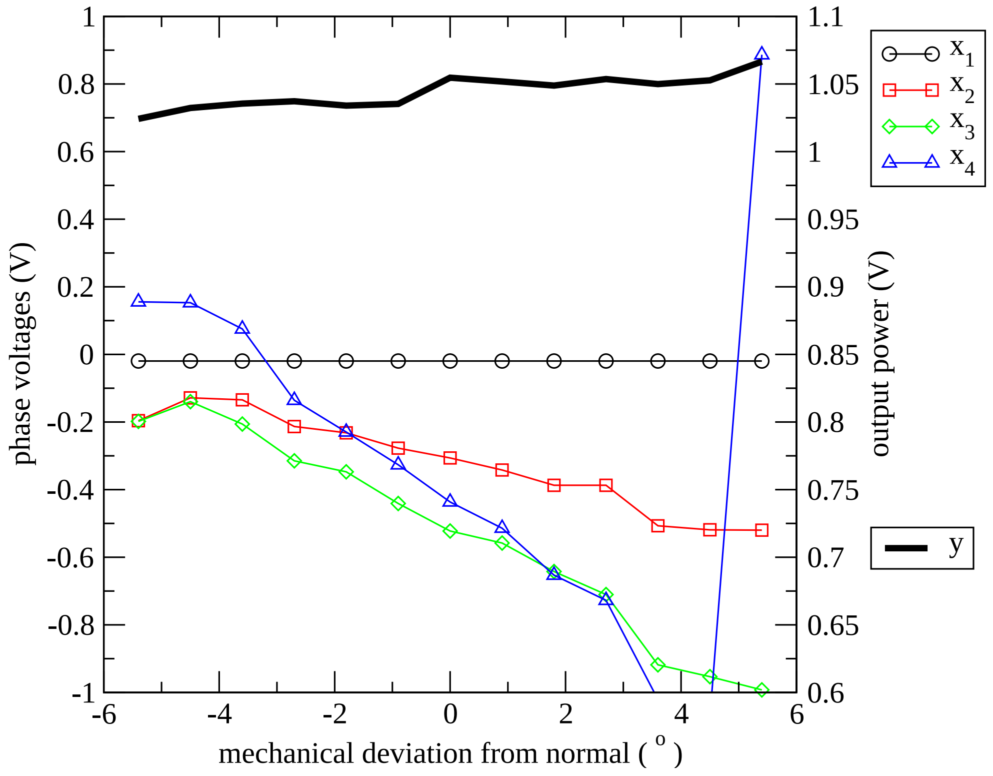


(b) Second test ten minutes after first test

Figure 4.8: Power vs direction with analog optimizer



(a) First test



(b) Second test ten minutes after first test

Figure 4.9: Power vs direction with digital optimizer

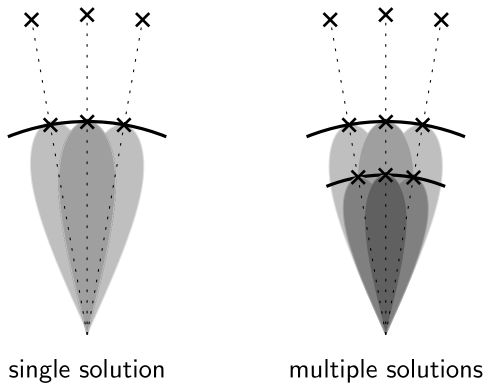


Figure 4.10: Multiple beamforming solutions

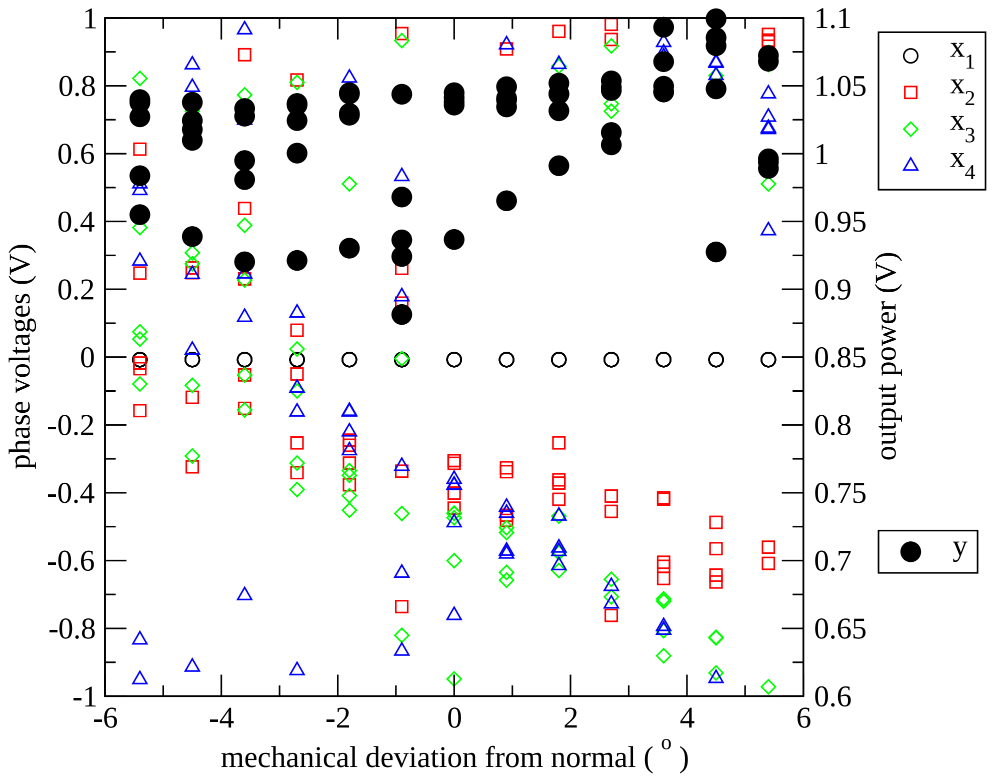


Figure 4.11: Power vs direction with analog optimizer, multiple trials

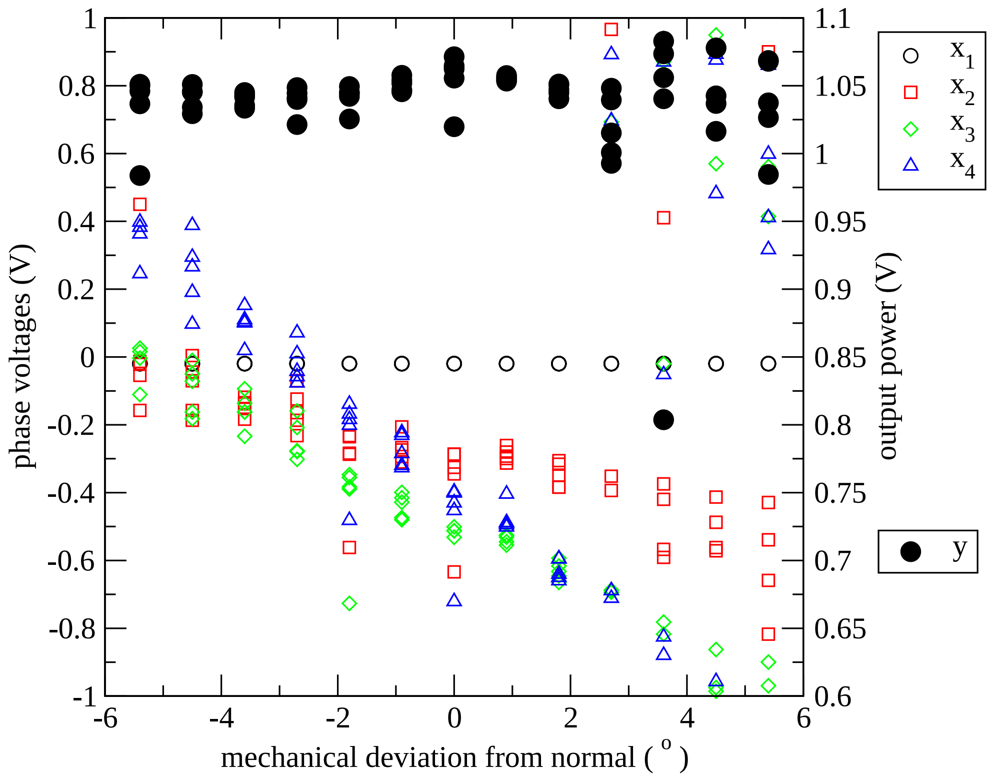


Figure 4.12: Power vs direction with digital optimizer, multiple trials

4.3.2 Transient tests

Transient experiments were performed to capture more detailed information about the optimization process. Here we are interested not only on the final solution after the algorithm has converged, but also the path taken by the weights during the optimization process, the convergence time, and any other irregularities such as overshoot and glitches. The first set of tests are performed with the simple test function because it is the most accurate method of comparing with the simulation results. The second set of tests are performed with the antenna function. All tests are performed with the analog algorithm only.

Test function tests

The first test analyses the transient response during a step change between two test functions. This test is a real world implementation of the simulation described in Section 3.7, Figure 3.12. Two test functions are created: $f_1(x_1, x_2) = -(x_1 + 0.6)^2 - (x_2 - 0.4)^2$ and $f_2(x_1, x_2) = -(x_1 - 0.3)^2 - (x_2 + 0.5)^2$. At time zero an instantaneous step occurs from $f_1(.)$ to $f_2(.)$, starting from an initial state where the algorithm is fully converged on the maximum of $f_1(.)$. The results show how the algorithm converges on the maximum of $f_2(.)$. The captured result appears in Figure 4.13. Repeatability was tested at the same time — many identical trials are performed and are superimposed on the plot in gray. One of the outcomes is highlighted and fully labeled. One needs to look very closely to see the other gray outcomes because they are

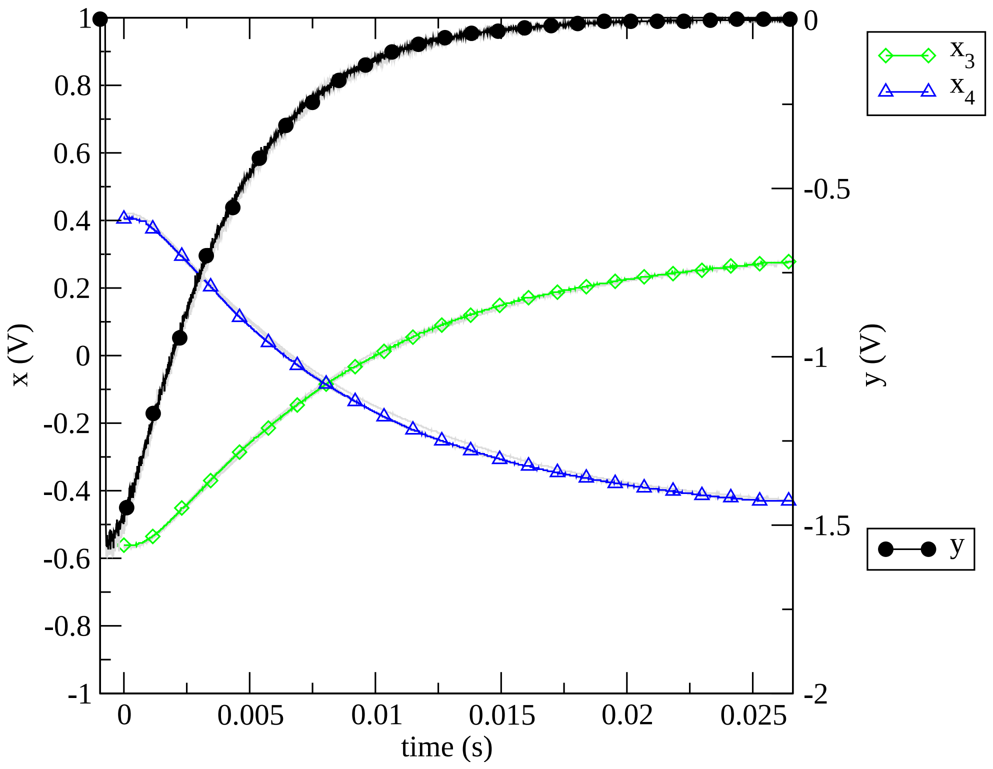


Figure 4.13: Test function transient response to fixed function step

almost perfectly superimposed, proof that the results are extremely repeatable. After almost 15 ms the function output has returned to within 90% of its maximum. This plot is remarkably similar to the simulation result in Figure 3.12.

In the next experiment the algorithm is tested with a series of random test functions $f_i(.)$. After the algorithm has achieved convergence on $f_k(.)$ a rapid step occurs to $f_{k+1}(.)$. This test is useful because it exercises the algorithm with many different possible test functions. The resulting function outputs are recorded in Figure 4.14. The convergence time is very similar in all cases. The algorithm achieves convergence in all test cases in less than 25 ms.

The next experiment tests for algorithm convergence with a single test function $f(.)$ but from random initial states. The weights are initialized with a random vector $\hat{\boldsymbol{x}}$ and the algorithm is turned on. After the weights have migrated to the final value $\boldsymbol{x}^\star$ which maximize $f(\hat{\boldsymbol{x}})$ the algorithm is stopped, the weights are reinitialized to a new random weight vector and the process repeats. Several outcomes from this experiment are recorded in Figure 4.15. The algorithm achieves convergence from all initial states tested within 25 ms.

Antenna function tests

The first experiment tests the algorithm convergence from random initial states. The antenna is pointed optimally, the phases are initialized with random values, and the algorithm is turned on. After convergence, the algorithm is stopped, the phases are reinitialized with new random values and the process is repeated. Figure 4.16 shows the received signal power in several recorded

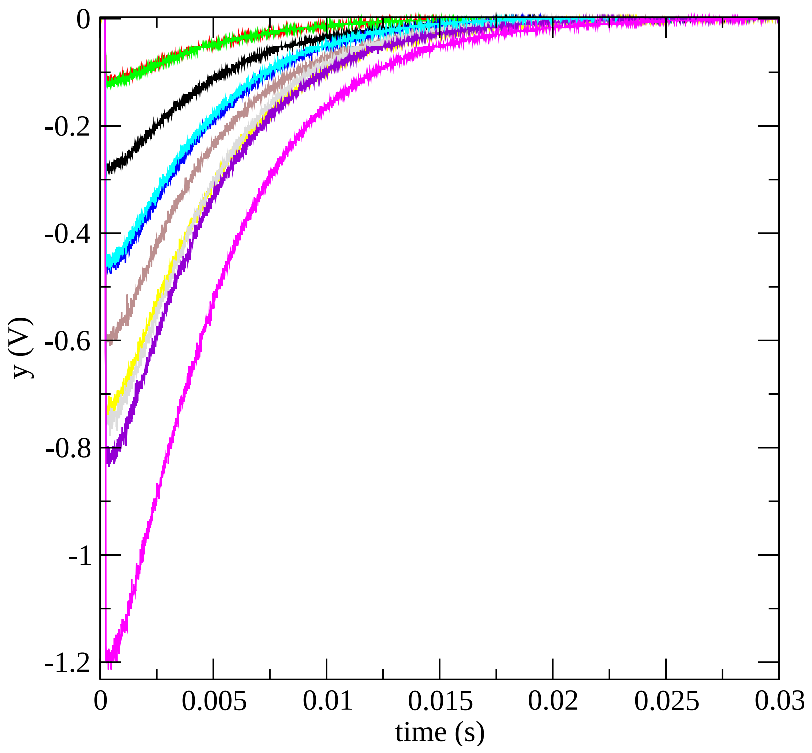


Figure 4.14: Test function transient response to random function step

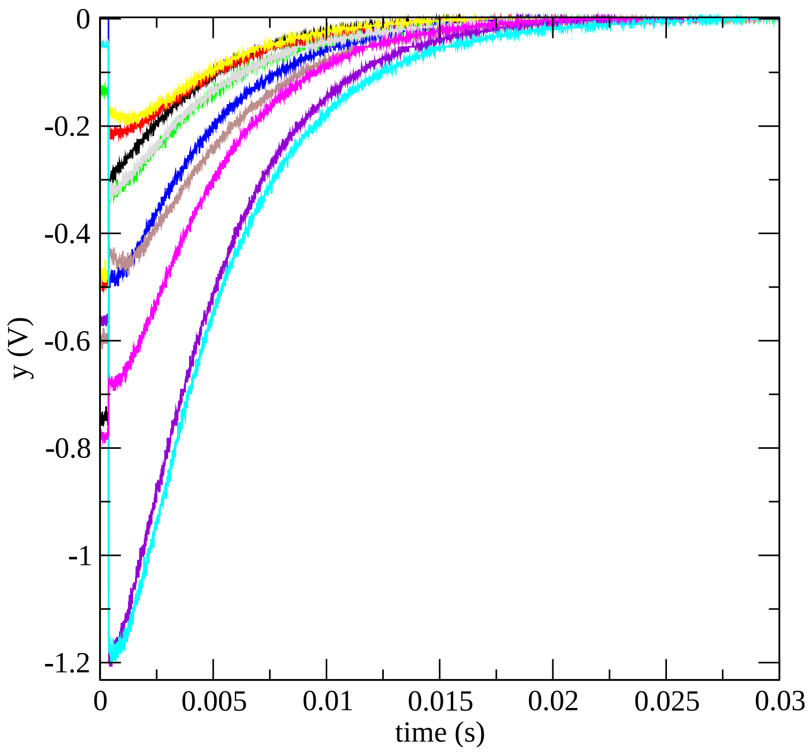


Figure 4.15: Test function transient response from random initial state

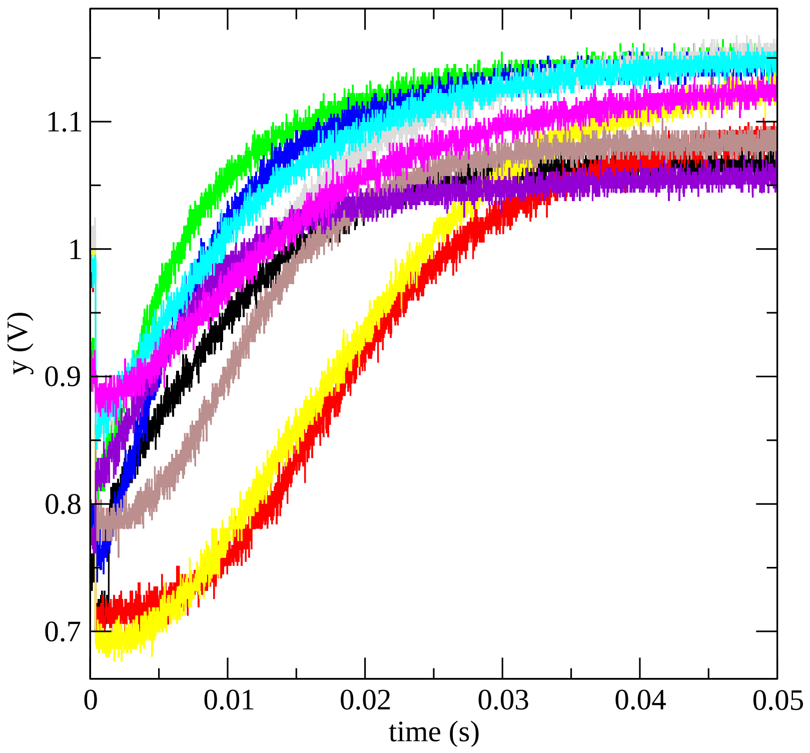


Figure 4.16: Antenna function transient response from random initial state

outcomes. It is apparent the the algorithm convergence time depends on the initial state. Furthermore, the final convergence value also depends on the initial state as can be confirmed by the experiment in Section 4.3.1, Figures 4.11 and 4.12. In all outcomes convergence occurs in less than approximately 50 ms. It is apparent that different initial states can produce very different results, making comparisons with simulation results difficult. As a rough comparison, a single outcome (Figure 4.17) is chosen from Figure 4.15 to have very similar characteristics with the simulation results in Figure 3.14.

The next experiment tests the algorithm convergence with a changing antenna function. There are not many degrees of freedom which can be practically controlled to change the antenna function. One practical method is to reorient the antenna using the stepper motor. The experiment description is as follows: orient the antenna optimally, turn on the algorithm and achieve convergence, then apply a rapid $6.3°$ mechanical deviation from normal using the stepper motor and watch the algorithm convergence. The test scenario is shown graphically in Figure 4.18. The results, including all phases and function output, are shown in Figure 4.19. For this test one would have preferred a true step input, but the mechanical dynamics do not permit this, resulting in a $6.3°$ step having a duration of approximately 100 ms. The algorithm manages to maintain an almost constant received signal power by adjusting the phases very rapidly (< 200 ms). The observed ringing is due to the mechanical vibrations caused by the intensity of the step, a phenomenon which was visible. The result of this experiment shows that the smart antenna can adapt its pattern to compensate for changes in orientation, the main goal of this work.

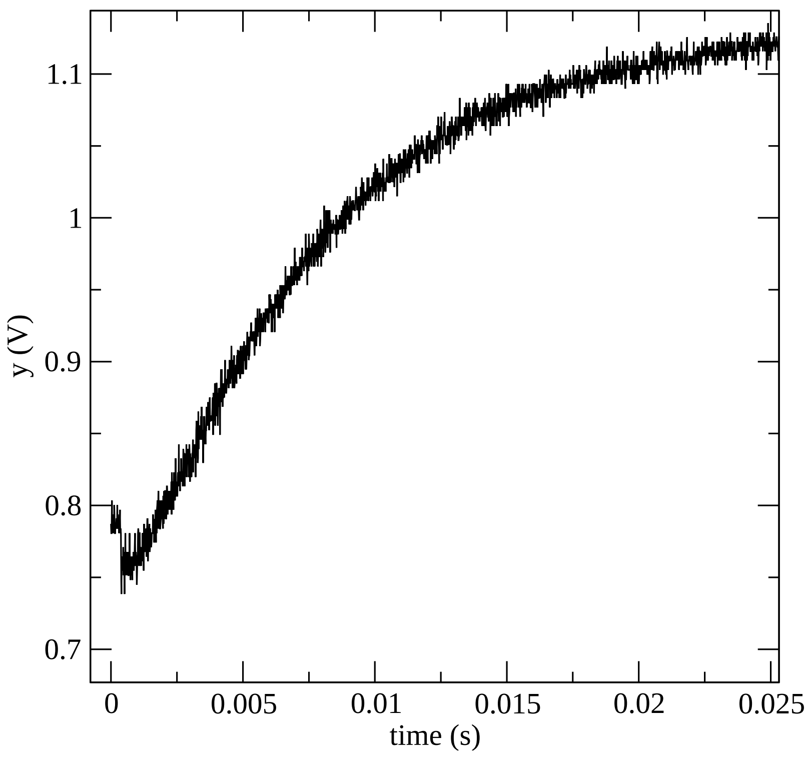


Figure 4.17: Antenna function transient response from random initial state

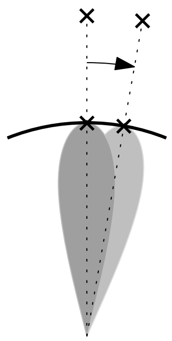


Figure 4.18: Antenna step reorientation

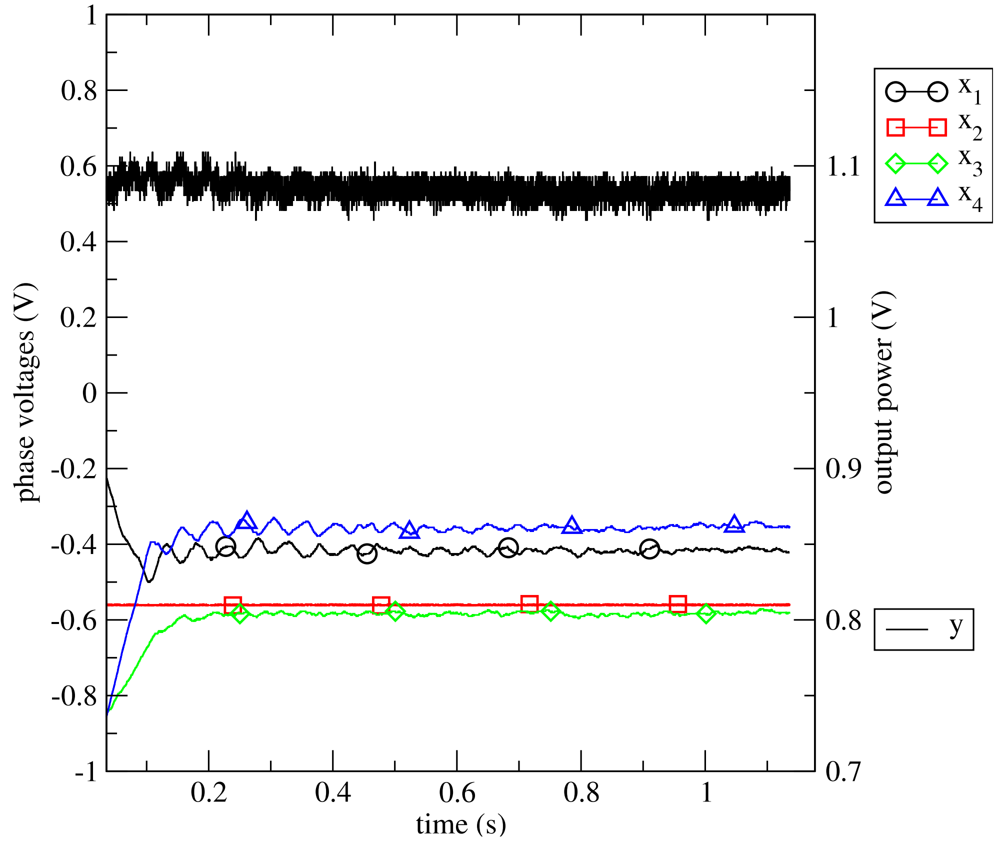


Figure 4.19: Antenna function transient response to motor $6.3°$ step

Chapter 5

Conclusions and future work

In this work a smart antenna system was implemented for the application of mobile digital satellite TV reception. A smart antenna system consists of (i) an antenna array having a set of weights which can be used to change the antenna pattern, (ii) a performance function which one would like the smart antenna to maximize, and (iii) an algorithm which continuously adjusts the weights to maximize the performance function. Stochastic approximation is a particularly attractive class of optimization algorithms because it does not require any knowledge of the function being optimized. A stochastic approximation algorithm continuously moves the current weight vector towards the optimum weight vector in the direction of the gradient. When the algorithm does not know the gradient it must estimate it. One way of achieving this is by applying small-amplitude simultaneous perturbations to the input vector and observing the effect on the function output.

Smart antenna algorithms are usually implemented in a software program running on a microprocessor. This requires a series of data conversion devices so that the digital discrete-time processor can interface with the analog continuous-time antenna circuit. In the interest of lower power consumption, cost and complexity, a continuous-time algorithm, implemented using only analog circuitry, is proposed for antenna beamforming. The algorithm performs a gradient descent operation using a gradient which is estimated by simultaneous perturbation. The proposed analog algorithm, although previously used in other applications such as learning of artificial neural networks, has not yet been applied to microwave antenna beamforming.

A circuit model of the smart antenna system is built and simulated using Agilent ADS. The optimizer circuit is then implemented using discrete components. This optimizer circuit is paired with a previously available antenna array and the resulting smart antenna is tested outdoors using a satellite signal. The results are in strong agreement with simulations and with another digitally implemented algorithm. Results show that the smart antenna system successfully adapts the antenna pattern to compensate for mechanical pointing errors.

Future efforts should be devoted to a more detailed theoretical analysis of the algorithm in order to accurately predict the effects of the design parameters on the algorithm performance. Using higher speed components would undoubtedly increase the algorithm performance, but of more interest are the stability and performance bounds achievable with a given technology. On the application side, we would like to pursue a VLSI implementation tailored to smart antennas, consisting of the analog algorithm as well as RF components.

Bibliography

[1] S. Safavi-Naeini, "Antennas and wireless systems, part 1", ECE476 Course notes, University of Waterloo, January 2000.

[2] J. R. Treichler and B. G. Agee, "A new approach to multipath correction of constant modulus signals", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, no. 2, pp. 459–472, April 1983.

[3] T. Ohira, "Adaptive array antenna beamforming architectures as viewed by a microwave circuit designer", in *Microwave Conference, 2000 Asia-Pacific*, December 2000, pp. 828–833.

[4] T. Ohira and K. Gyoda, "Electronically steerable passive array radiator antennas for low-cost analog adaptive beamforming", in *IEEE International Conference on Phased Array Systems and Technology*, 2000, pp. 101–104.

[5] B. Widrow, P. E. Mantey, L. J. Griffiths, and B. B. Goode, "Adaptive antenna systems", *Proceedings of the IEEE*, vol. 55, no. 12, pp. 2143–2159, 1967.

[6] James C. Spall, *Introduction to Stochastic Search and Optimization*, John Wiley & Sons, Inc., New York, NY, USA, 2003.

[7] A. Dembo and T. Kailath, "Model-free distributed learning", *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 58–70, March 1990.

[8] G. Cauwenberghs, "A fast stochastic error-descent algorithm for supervised learning and optimization", in *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, San Francisco, CA, USA, 1993, pp. 244–251, Morgan Kaufmann Publishers Inc.

[9] G. Cauwenberghs, "An analog vlsi recurrent neural network learning a continuous-time trajectory", *IEEE Transactions on Neural Networks*, vol. 7, no. 2, pp. 346–361, March 1996.

[10] J. Alspector, R. Meir, B. Yuhas, A. Jayakumar, and D. Lippe, "A parallel gradient descent method for learning in analog vlsi neural networks", in *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, San Francisco, CA, USA, 1993, pp. 836–844, Morgan Kaufmann Publishers Inc.

[11] D. B. Kirk, D. Kerns, K. Fleischer, and A. H. Barr, “Analog vlsi implementation of multi-dimensional gradient descent”, in *Neural Information Processing Systems*, 1993, vol. 5, pp. 789–796.

[12] Y. Maeda, H. Hirano, and Y. Kanata, “A learning rule of neural networks via simultaneous perturbation and its hardware implementation”, *Neural Networks*, vol. 8, no. 2, pp. 251–259, 1995.

[13] A. C. Carusone and D. A. Johns, “Analog filter adaptation using a dithered linear search algorithm”, in *ISCAS (4)*, 2002, pp. 269–272.

[14] J. Alspector, J. W. Gannett, S. Haber, M. B. Parker, and R. Chu, “A vlsi-efficient technique for generating multiple uncorrelated noise sources and its application to stochastic neural networks”, *IEEE Transactions on Circuits and Systems*, vol. 38, no. 1, pp. 109–123, January 1991.

[15] J. Cheng, Y. Kamiya, and T. Ohira, “Adaptive beamforming of espar antenna using sequential perturbation”, in *Microwave Symposium Digest, 2001 IEEE MTT-S International*, 2001, vol. 1, pp. 133–136.

[16] C. Sun, A. Hirata, T. Ohira, and N. C. Karmakar, “Fast beamforming of electronically steerable parasitic array radiator antennas: Theory and experiment”, *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 7, pp. 1819–1832, July 2004.

[17] James C. Spall, “A one-measurement form of simultaneous perturbation stochastic approximation”, *Automatica*, vol. 33, no. 1, pp. 109–112, 1997.

[18] Y. Maeda, “Time difference simultaneous perturbation method”, *Electronics Letters*, vol. 32, pp. 1016–1018, 1996.

[19] B. Widrow and J. Mccool, “A comparison of adaptive algorithms based on the methods of steepest descent and random search”, *IEEE Transactions on Antennas and Propagation*, vol. 24, no. 5, pp. 615–637, 1976.

[20] B. P. Anderson, “Low-pass filters as expectation operators for multiplicative noise”, *IEEE Transactions on Circuits and Systems*, vol. 39, no. 12, pp. 871–873, December 1992.

[21] David M. Pozar, *Microwave engineering*, John Wiley & Sons, Inc., New York, NY, USA, 1998.

Appendix A

Simulation schematics

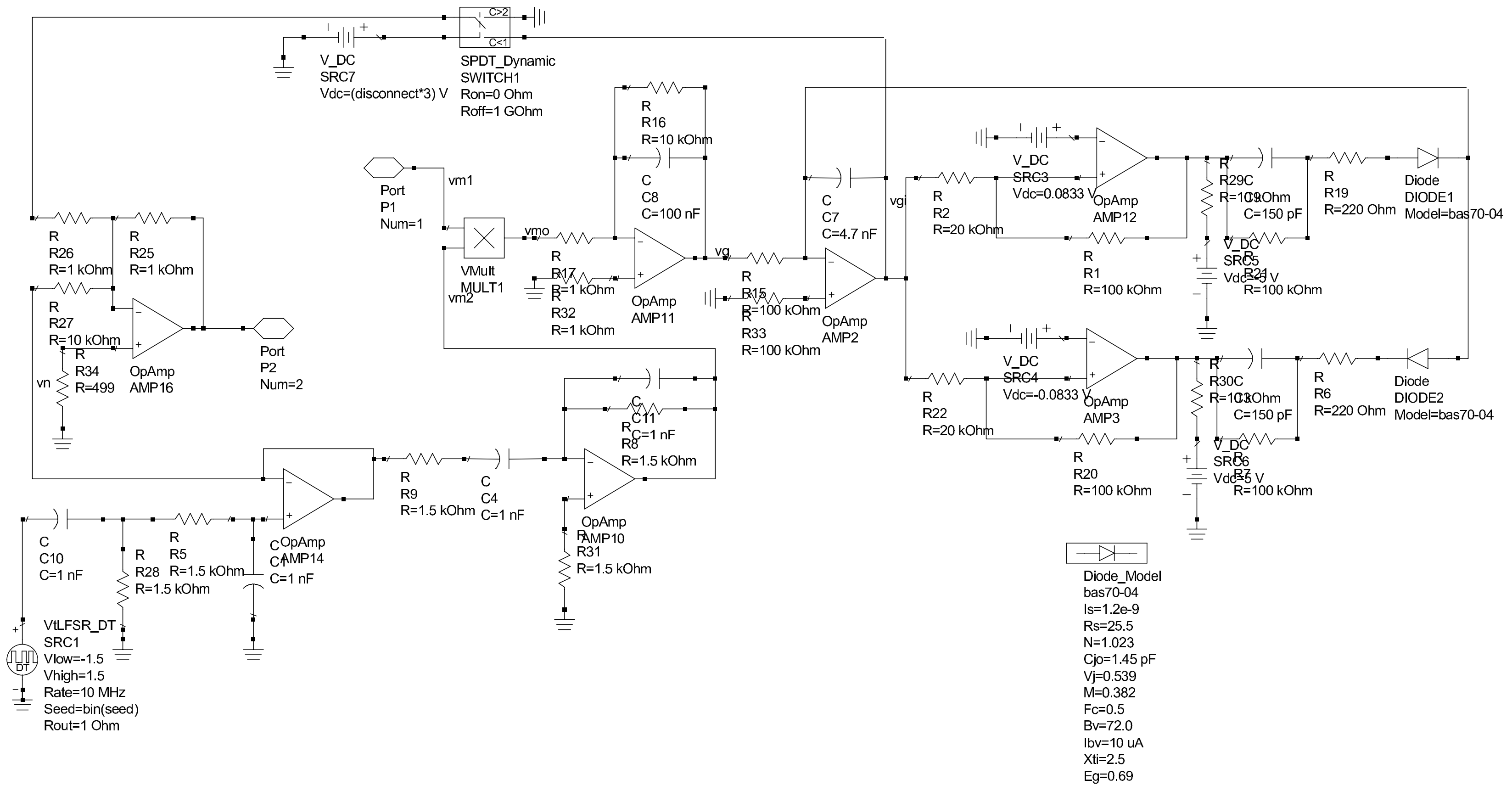
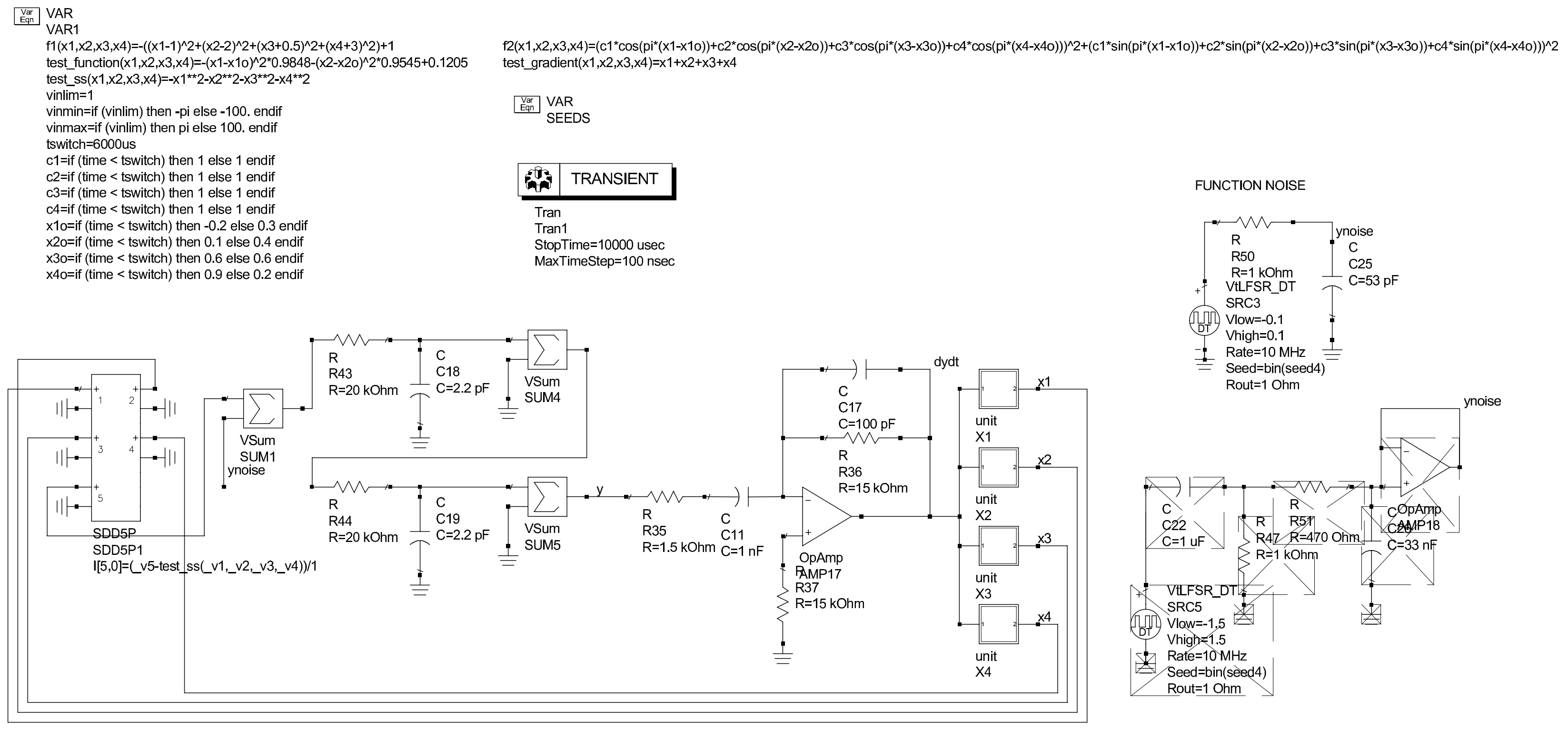


Figure A.1: Simulation schematics: single optimizer unit



```
VAR
VAR1
f1(x1,x2,x3,x4)=-((x1-1)^2+(x2-2)^2+(x3+0.5)^2+(x4+3)^2)+1
test_function(x1,x2,x3,x4)=-(x1-x1o)^2*0.9848-(x2-x2o)^2*0.9545+0.1205
test_ss(x1,x2,x3,x4)=-x1**2-x2**2-x3**2-x4**2
vinlim=1
vinmin=if (vinlim) then -pi else -100. endif
vinmax=if (vinlim) then pi else 100. endif
tswitch=6000us
c1=if (time < tswitch) then 1 else 1 endif
c2=if (time < tswitch) then 1 else 1 endif
c3=if (time < tswitch) then 1 else 1 endif
c4=if (time < tswitch) then 1 else 1 endif
x1o=if (time < tswitch) then -0.2 else 0.3 endif
x2o=if (time < tswitch) then 0.1 else 0.4 endif
x3o=if (time < tswitch) then 0.6 else 0.6 endif
x4o=if (time < tswitch) then 0.9 else 0.2 endif

f2(x1,x2,x3,x4)=(c1*cos(pi*(x1-x1o))+c2*cos(pi*(x2-x2o))+c3*cos(pi*(x3-x3o))+c4*cos(pi*(x4-x4o)))^2+(c1*sin(pi*(x1-x1o))+c2*sin(pi*(x2-x2o))+c3*sin(pi*(x3-x3o))+c4*sin(pi*(x4-x4o)))^2
test_gradient(x1,x2,x3,x4)=x1+x2+x3+x4

VAR
SEEDS

TRANSIENT
Tran
Tran1
StopTime=10000 usec
MaxTimeStep=100 nsec

SDD5P
SDD5P1
I[5,0]=(_v5-test_ss(_v1,_v2,_v3,_v4))/1
```

Figure A.2: Simulation schematics: top level

Appendix B

Optimizer implementation schematics

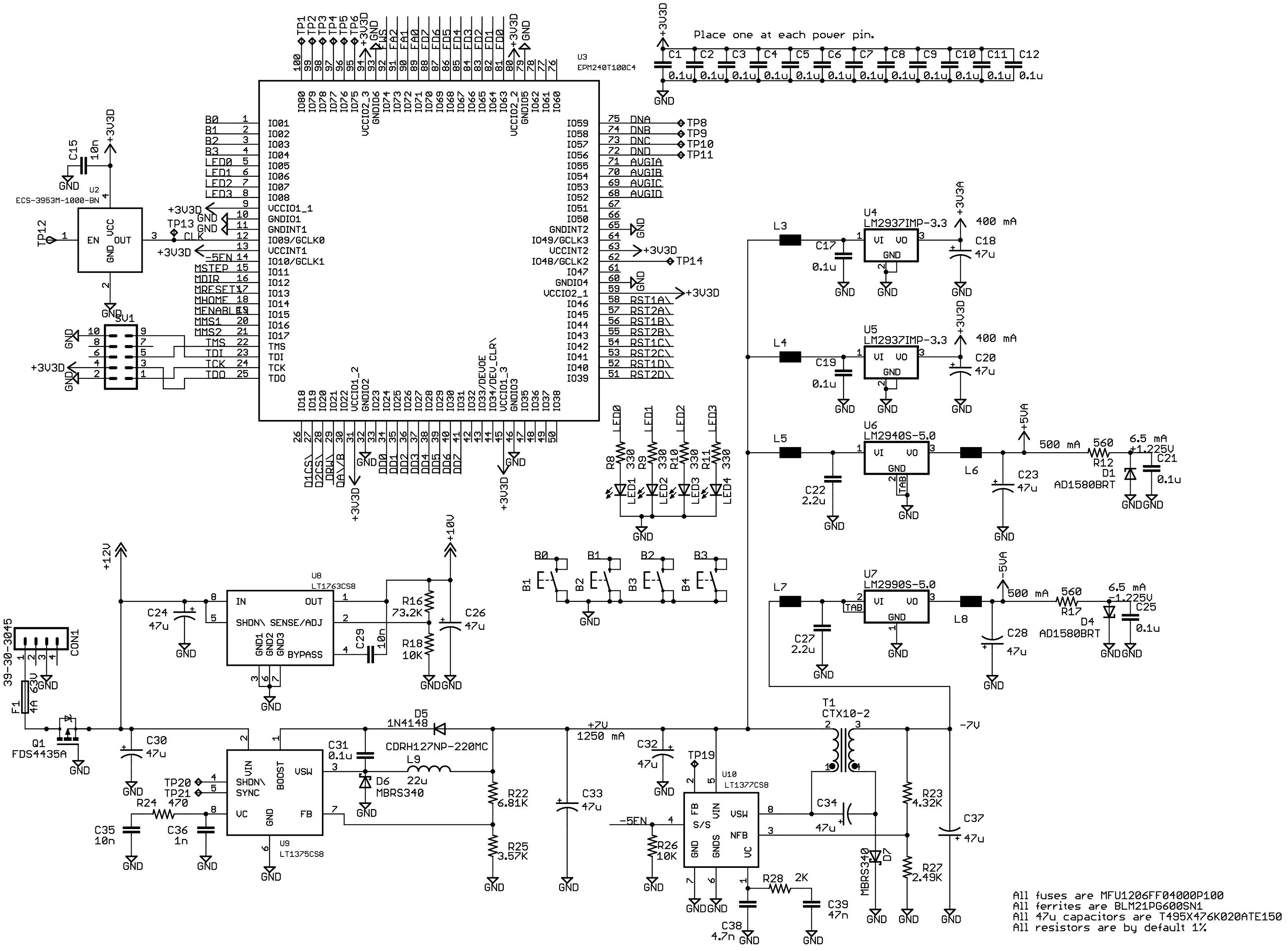


Figure B.1: Implementation schematics page 1/9

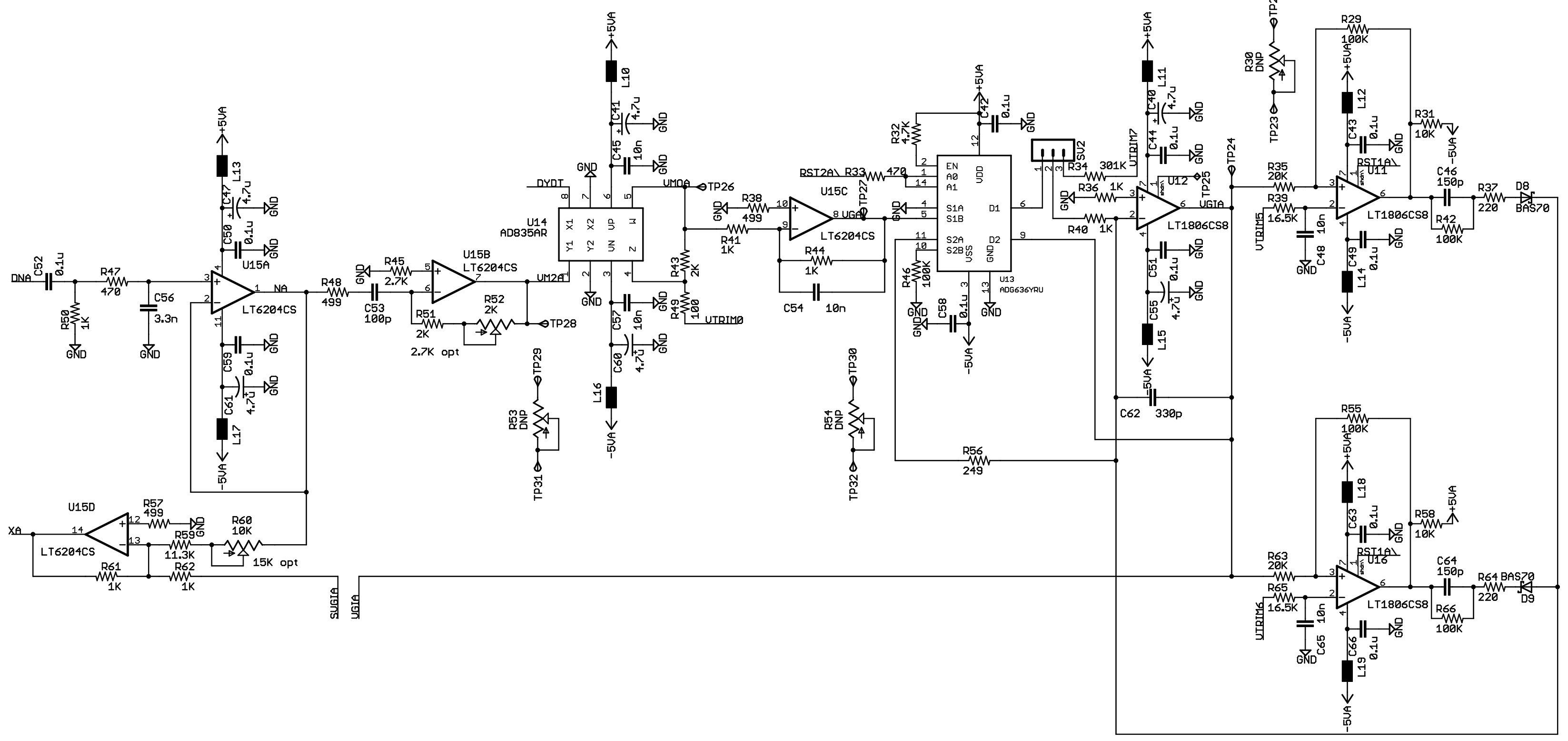


Figure B.2: Implementation schematics page 2/9

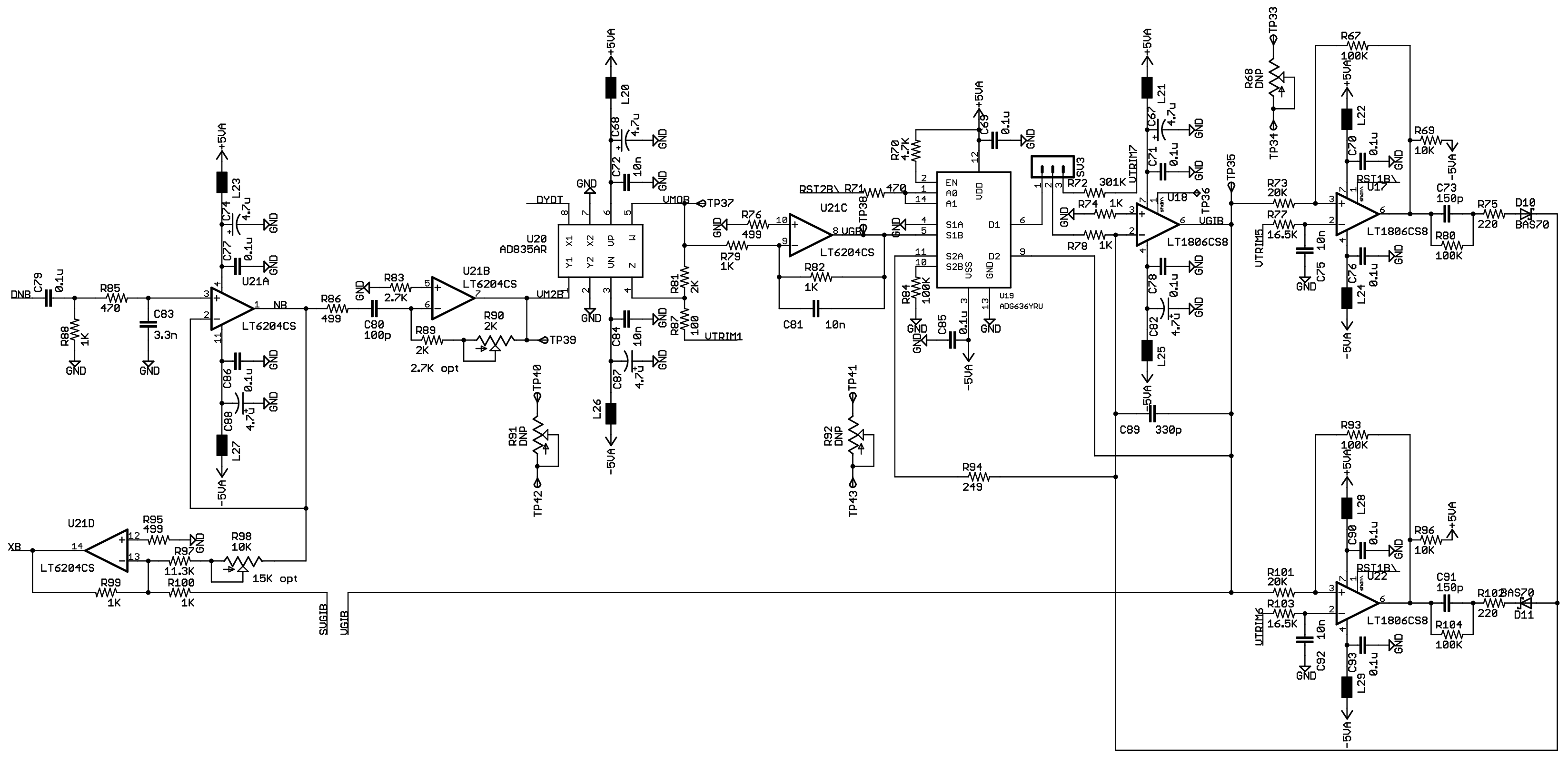


Figure B.3: Implementation schematics page 3/9

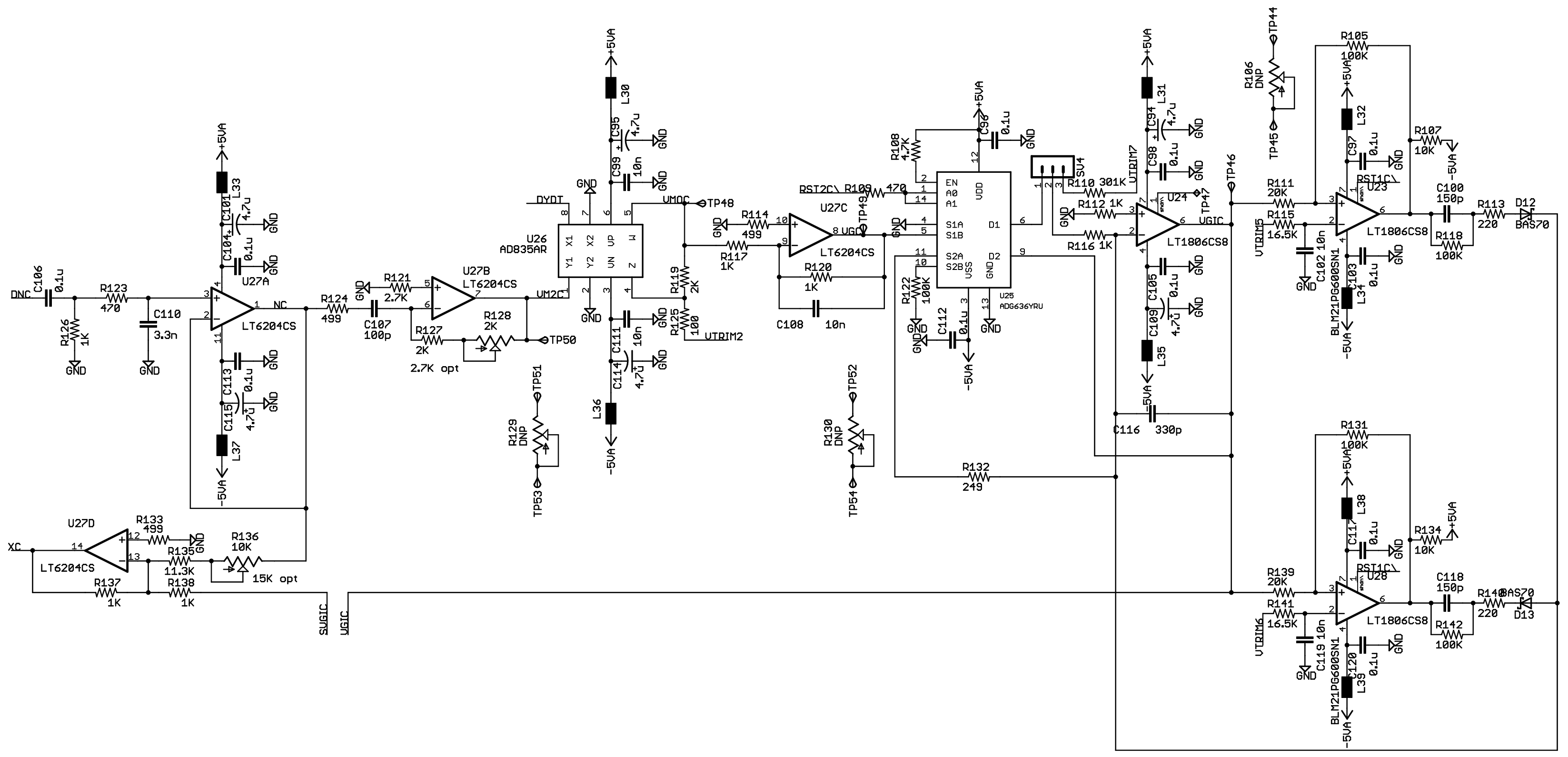


Figure B.4: Implementation schematics page 4/9

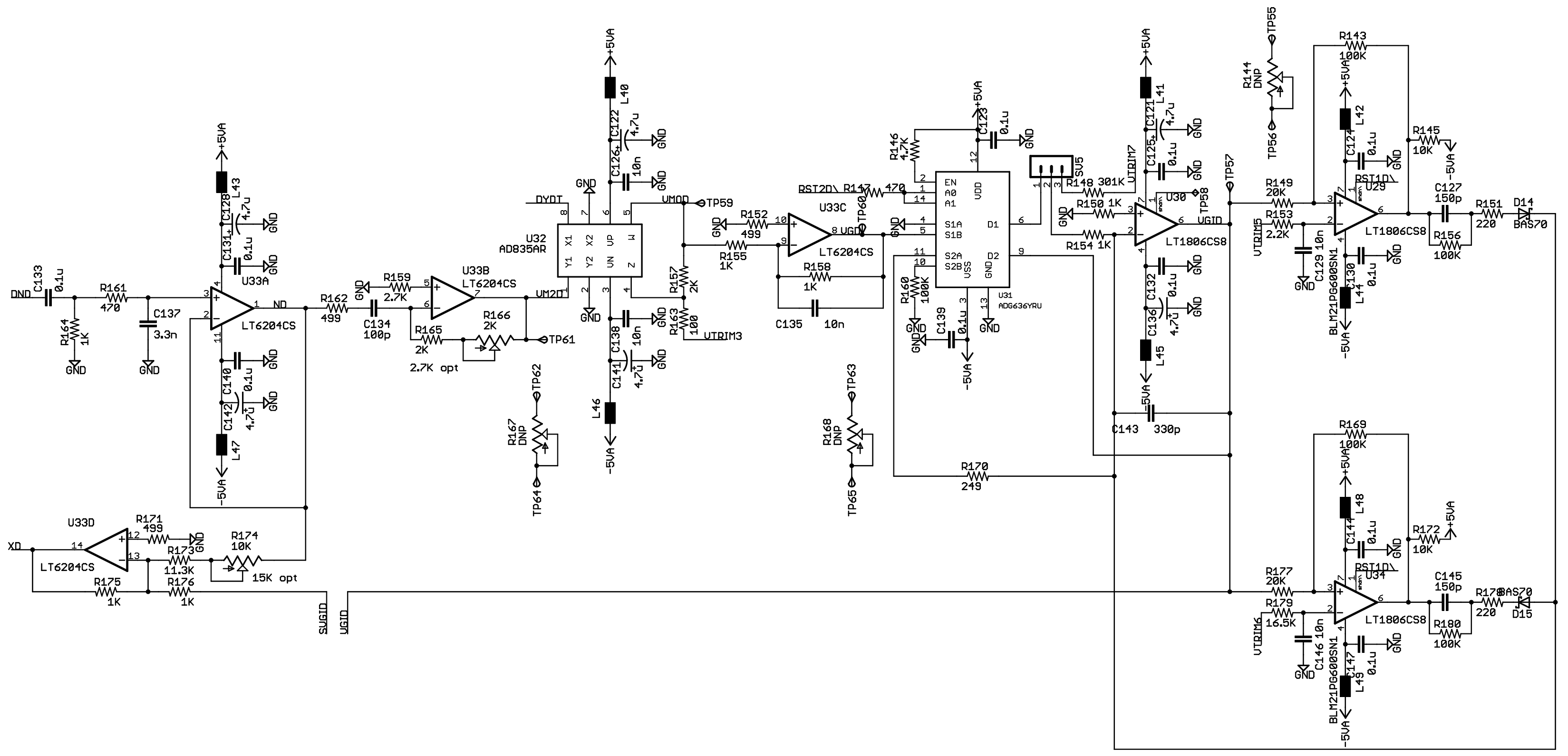


Figure B.5: Implementation schematics page 5/9

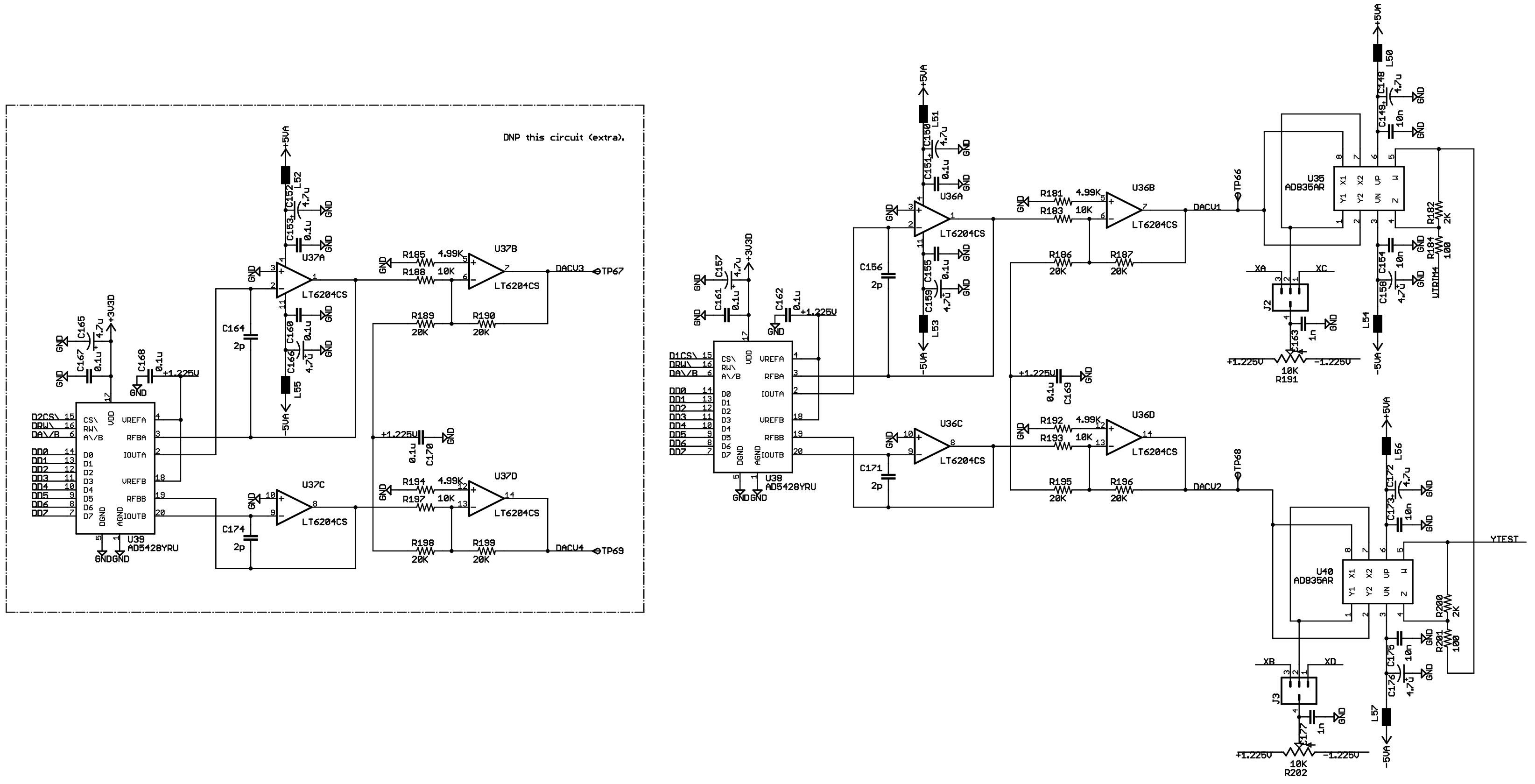


Figure B.6: Implementation schematics page 6/9

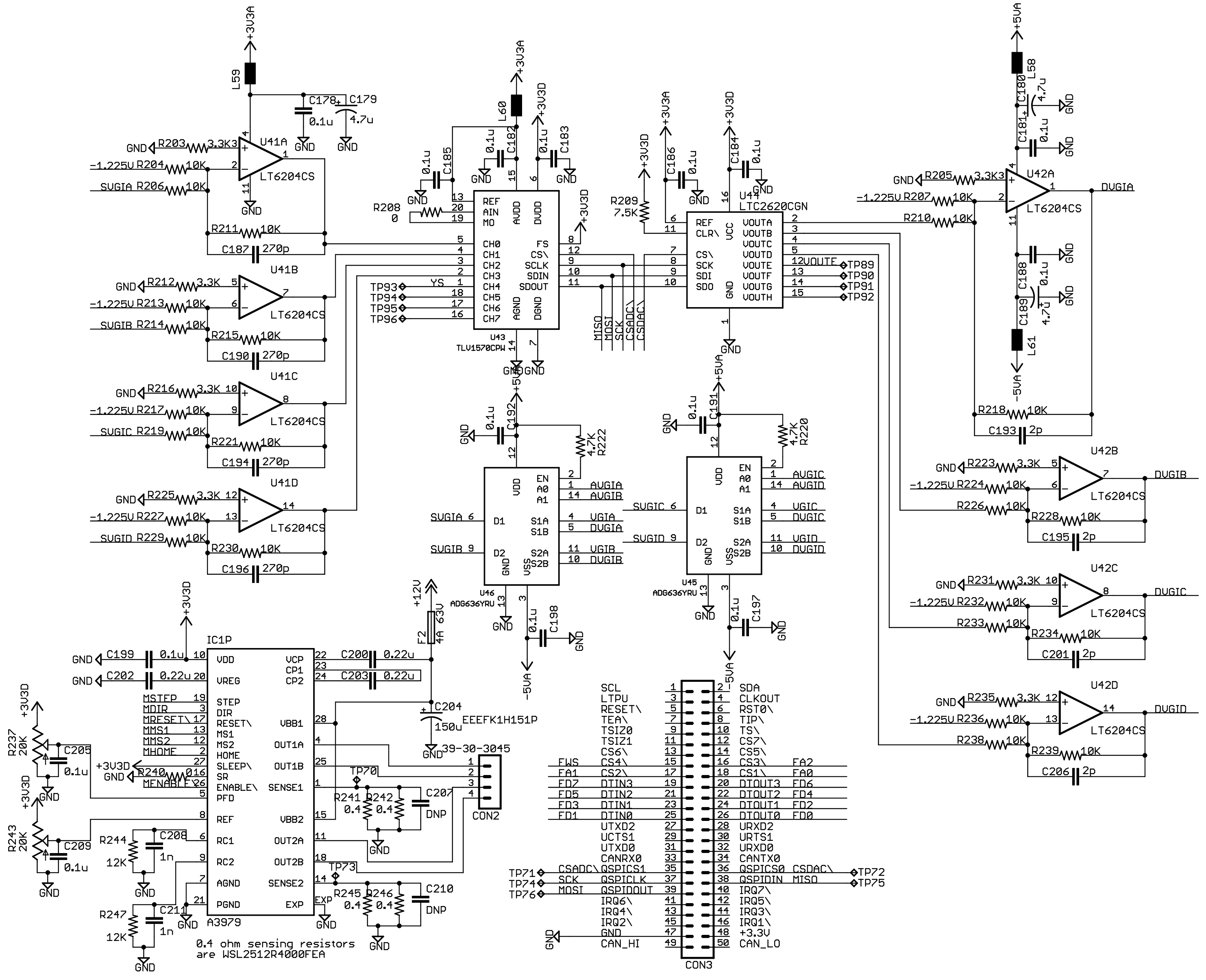


Figure B.7: Implementation schematics page 7/9

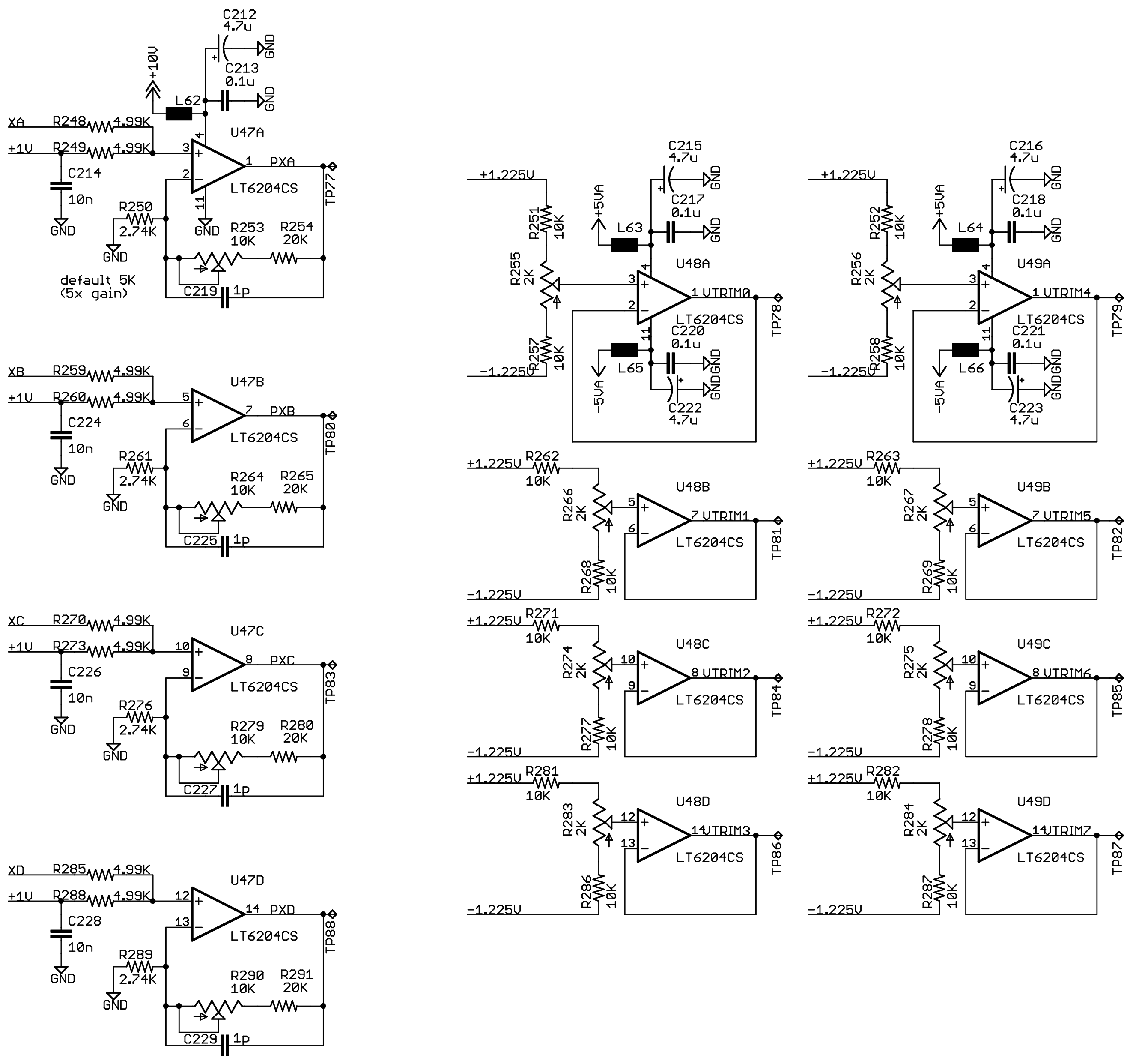


Figure B.8: Implementation schematics page 8/9

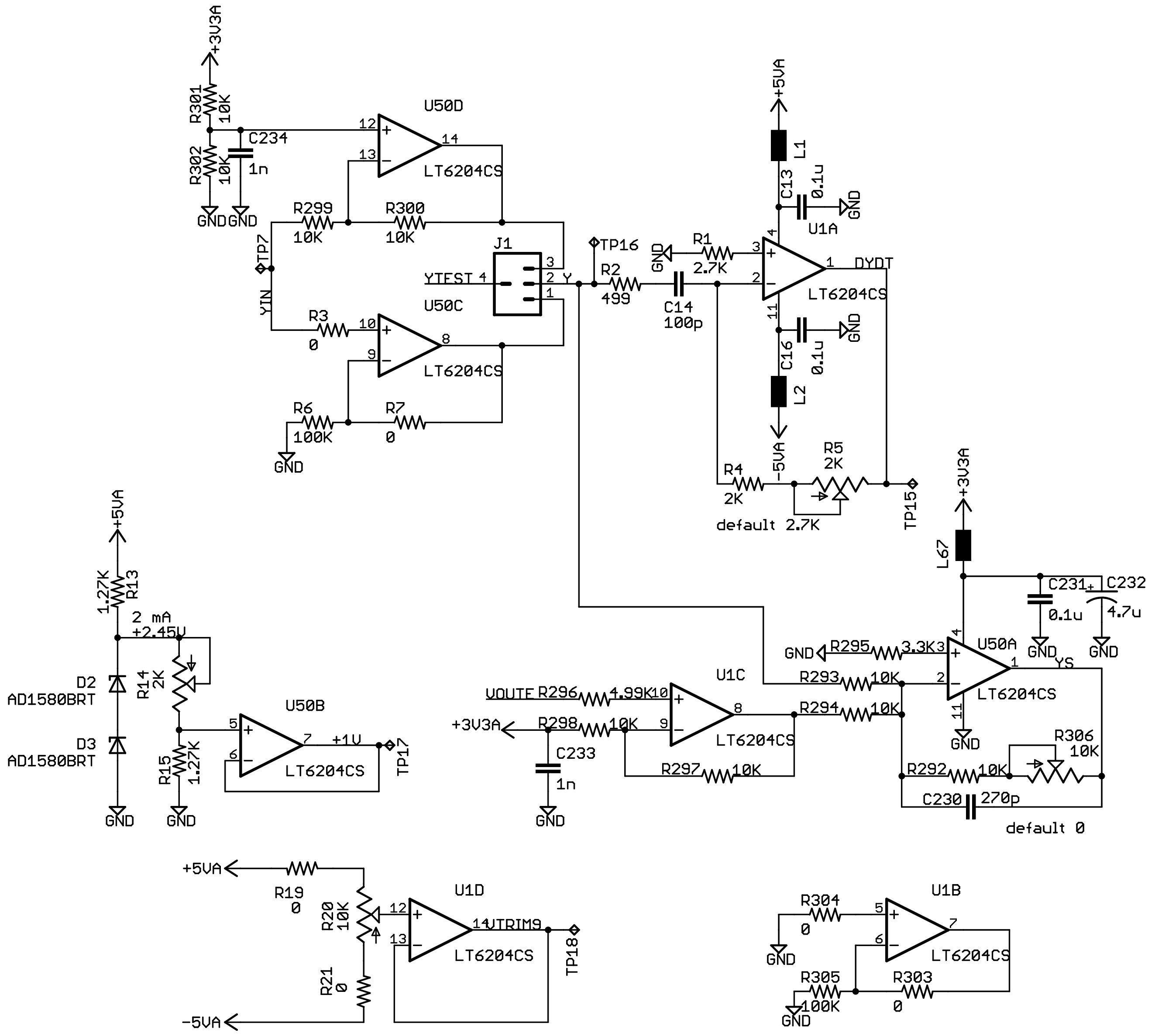


Figure B.9: Implementation schematics page 9/9