UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DTIC FILE COPY

1

AD-A196 647

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER<br>AFIT/CI/NR 88- 54 | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle)<br>ANALYSIS AND DESIGN OF INTERCONNECTION NETWORKS FOR PHASED ARRAY ANTENNAS | | 5. TYPE OF REPORT & PERIOD COVERED<br>MS THESIS |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| AUTHOR(s)<br>JOSEPH D. DOWNS, II | | 8. CONTRACT OR GRANT NUMBER(s) |
| PERFORMING ORGANIZATION NAME AND ADDRESS<br>AFIT STUDENT AT: UNIVERSITY OF LOUISVILLE | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| CONTROLLING OFFICE NAME AND ADDRESS | | 12. REPORT DATE<br>1988 |
| | | 13. NUMBER OF PAGES<br>222 |
| MONITORING AGENCY NAME & ADDRESS (If different from Controlling Office)<br>AFIT/NR<br>Wright-Patterson AFB OH 45433-6583 | | 15. SECURITY CLASS. (of this report)<br>UNCLASSIFIED |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

DISTRIBUTED UNLIMITED: APPROVED FOR PUBLIC RELEASE

DTIC ELECTE AUG 0 2 1988 S D H

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

SAME AS REPORT

18. SUPPLEMENTARY NOTES

Approved for Public Release: IAW AFR 190-1
LYNN E. WOLAVER [illegible] 19 July 88
Dean for Research and Professional Development
Air Force Institute of Technology
Wright-Patterson AFB OH 45433-6583

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

[redacted]

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

PII Redacted

## ABSTRACT

This thesis explores the possibility of designing an electronically steerable phased array antenna system using a LC impedance matching network and sections of transmissiom line as an interconnection medium between pairs of radiating elements. This interconnection network would be used to control the direction of the array's radiation pattern. Adjustments of the parameters of the LC network would attempt to enforce a desired current distribution on the elements of the array, resulting in the desired radiation pattern.

Two design procedures have been investigated and are discussed with presentation of results and sample radiation patterns obtained. The first design method uses a Taylor's series expansion to linearize the network equations describing the array. The other design method utilizes an optimization routine to systematically adjust the parameters of the impedance matching networks until the desired current distributions are realized as closely as possible.

# ANALYSIS AND DESIGN OF INTERCONNECTION NETWORKS FOR PHASED ARRAY ANTENNAS

By

Joseph D. Downs II
B.S., University of Louisville, 1985

A Thesis
Submitted to the Faculty of the
University of Louisville
Speed Scientific School
as Partial Fulfillment of the Requirements
for the Professional Degree

MASTER OF ENGINEERING

Department of Electrical Engineering

January 1987

ANALYSIS AND DESIGN
OF INTERCONNECTION NETWORKS
FOR PHASED ARRAY ANTENNAS.

Submitted by: Joseph D. Downs II
Joseph D. Downs II

A Thesis Approved on

May 1, 1987
Date

by the Following Reading and Examination Committee:

[illegible]
Thesis Director, J. Carroll Hill

[illegible]
Donald J. Scheer

[illegible]
Thomas L. Holloman

W. L. KO
Wai L. Ko

ACKNOWLEDGEMENTS

The author would like to whole heartily thank his thesis advisor, Dr. J.C. Hill for his instruction and ideas concerning the topics of this thesis. Appreciation is extended to the members of the thesis committee for their work in reading the final thesis document.

Much thanks and appreciation are also extended to the author's brother, Mr. Woodrow F. Downs, for his help in editing and proof reading the text of this thesis. The author's father, Mr. Joseph D. Downs provided much assistance with problems encountered in learning and using APL on the IBM PC during the early development of the design procedures in this thesis.

To the author's wife, Julie Ritchie-Downs, the highest thanks and appreciation are extended for her enduring love and support during the year and a half of research encompassing this thesis. Her understanding and helpfulness was a great relief to the author during his investigations.

TABLE OF CONTENTS

LIST OF FIGURES

# I. INTRODUCTION

An antenna is a device used in the transmission and reception of electromagnetic energy. Sometimes, as is the case with most broadcast radio stations, transmission is done most effectively and economically with a single radiating element. However, when an application requires higher gain, directivity, steerability, or other characteristics that a single element cannot provide, an antenna array consisting of several radiating elements may be necessary to satisfy one's requirements. In the field of amateur radio, phased array antennas are used quite frequently to provide the user with the type of radiation pattern desired. Also, the military has used large antenna arrays for defense against ballistic missiles and in the surveillance and tracking of objects in space.

Another group that has recently become interested in the use of phased arrays is the air traffic control community. Since World War II, simple mechanically steered antennas have been used for air traffic control, but with the enormous increase in the number of takeoffs and landings, types of aircraft, and increase in their speeds, the potential use for computer controlled phased arrays is quite evident.[1]

In an antenna array consisting of a single type of radiating element, there are generally four parameters that can be adjusted: the number of radiating elements, the spatial distribution of the elements, the amplitude, and the phase of the currents used to excite the elements.

When considering the synthesis of a phased array antenna system, there are two basic requirements that must be met. First, determination of the current distribution on each element of the array that will produce the desired radiation pattern must be done. The second task, which is extensively explored in this thesis, is the determination of the proper impedance transforming and phase shifting networks needed to produce the desired current distribution on the array. Traditionally, in the simple arrays found in broadcast and amateur radio, the impedance transforming and phase shifting medium is a transmission line connected between the individual array element and the transmitter.

In this thesis, a method for calculating the current distribution for a desired radiation pattern was used as a starting point for the development of the interconnection networks.[2] This procedure allows a user to specify a direction of maximum radiation (main lobe) and one or more directions with a minimum amount of radiation (nulls). This method of beam steering was adapted to the triangular geometry of an array consisting of three equally spaced radiating elements as shown in

Figure 1. The number of steerable nulls is dependent upon the number of elements in the array: the number of nulls is equal to half the number of elements. Therefore, the radiation patterns in this thesis will be specified by one null direction and a main lobe direction.

As an example of the beam steering procedure, a radiation pattern and the corresponding current excitation on the elements is presented in Figure 2. This pattern was generated from an input of $\Theta_m = 180$ degrees for the direction of maximum radiation and $\Theta_n = 320$ degrees for the null direction. The radiation pattern shown in Figure 3 was generated by a 5 element array with the direction of maximim radiation at 50 degrees and the two nulls specified at +180 degrees and 270 degrees.[3] Note that additional nulls may be produced in unspecified directions.

This thesis will explore an unconventional method of enforcing the desired current distribution on the array. The elements are interconnected in pairs, using LC impedance matching networks and short sections of transmission line as an interconnection medium between pairs of radiating elements. The LC impedance matching network is in the form of a PI network which is assumed lossless. This PI network has capacitors in the two parallel branches and an inductor in the series branch, as shown in Figure 4. Figure 5 shows the 3 radiating

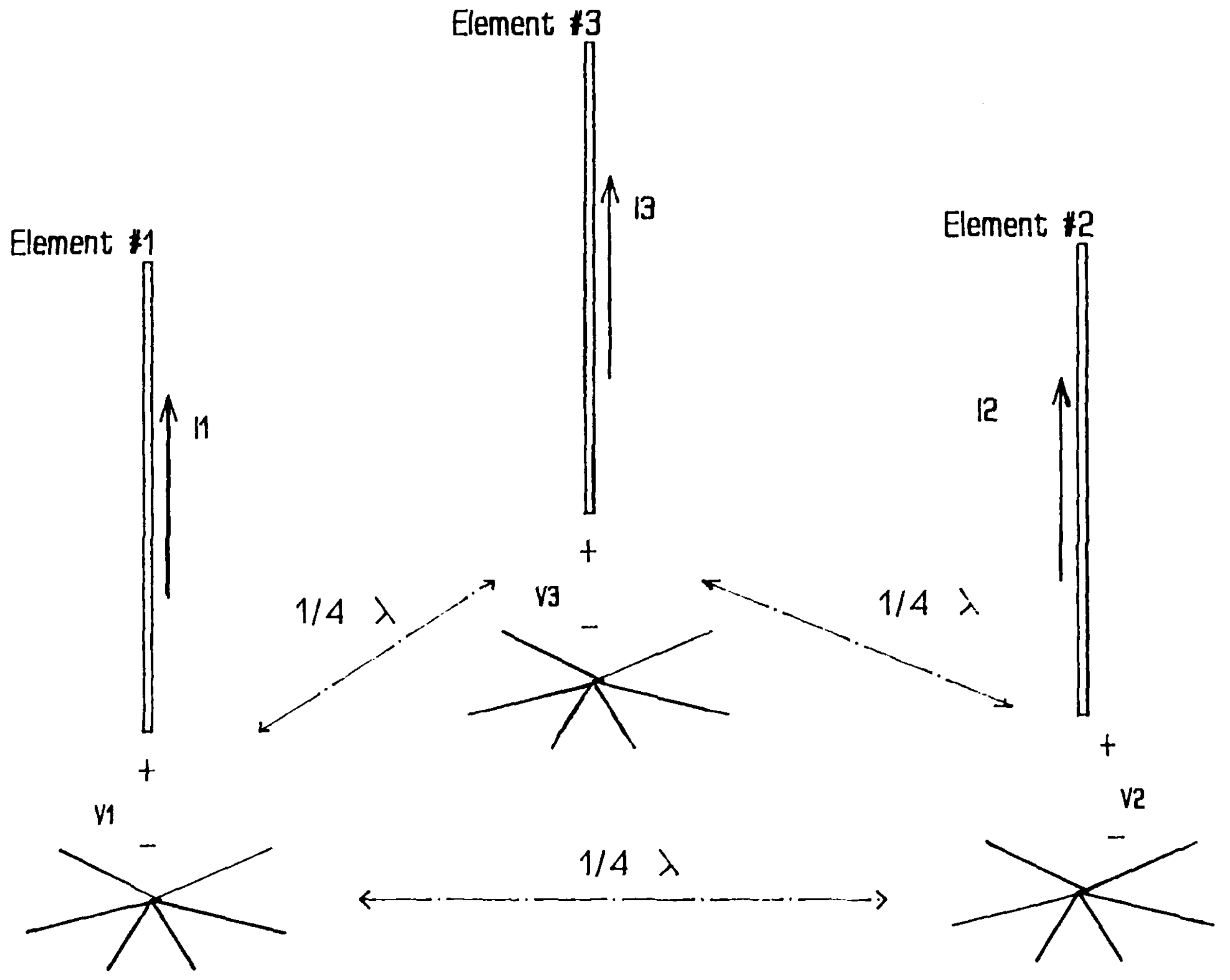
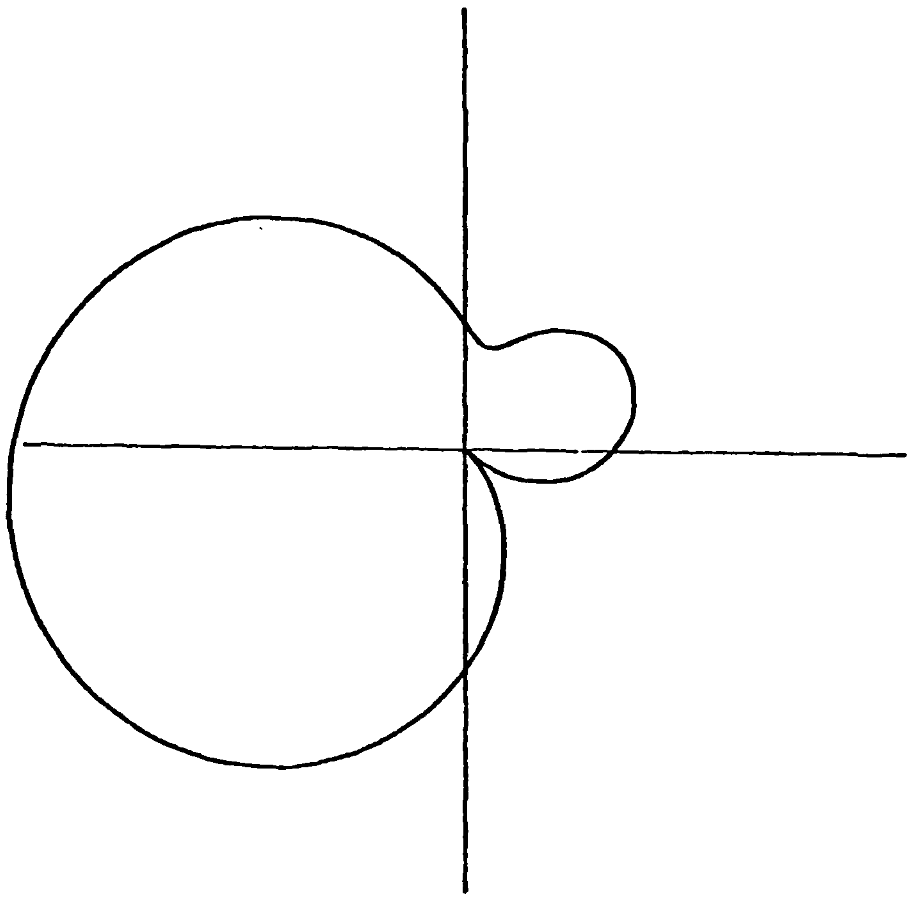


FIGURE 1 - Three Element Triangular Array



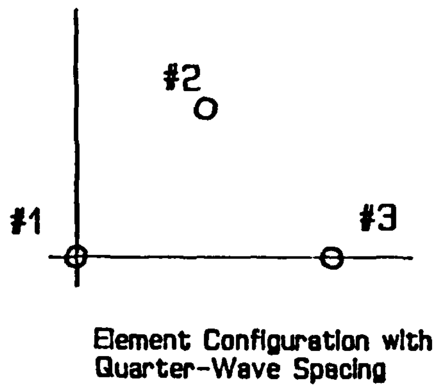
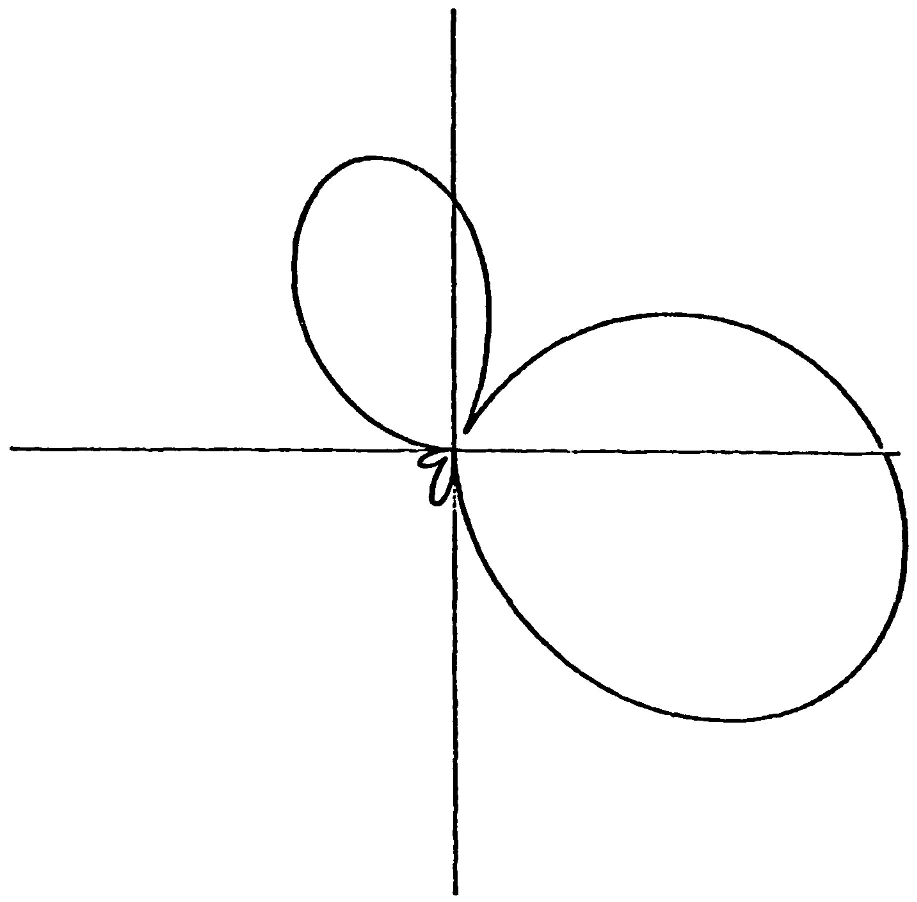
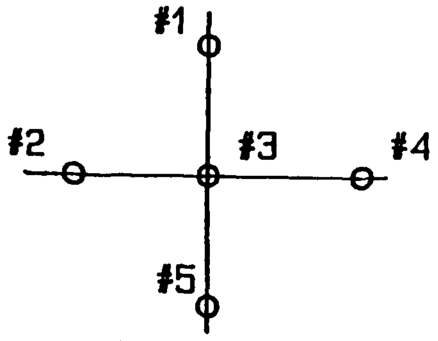


FIGURE 2 - Three Element Array Pattern

Om = 180 and On = 320





Element Configuration with
Quarter-Wave Spacing

FIGURE 3 - Five Element Array Pattern

$\theta m = 320,\ \theta n = 180,$ and $On = 270$

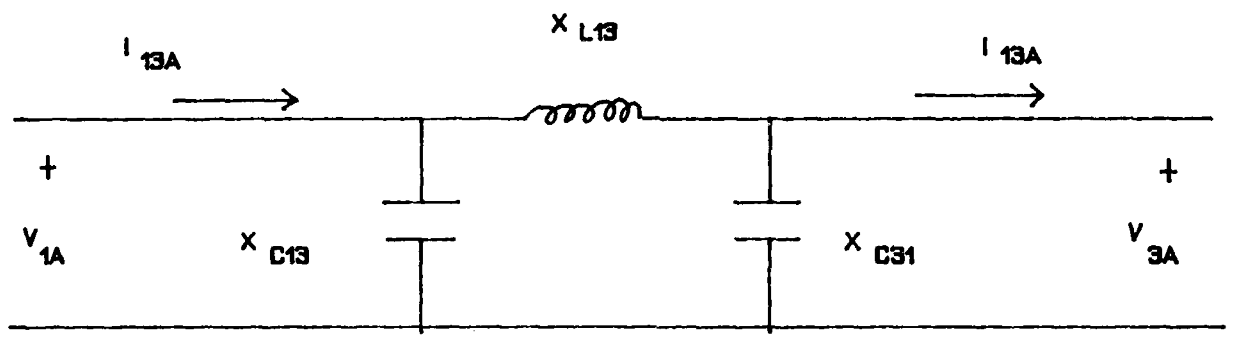


FIGURE 4 - PI Network of Phased Array Antenna System

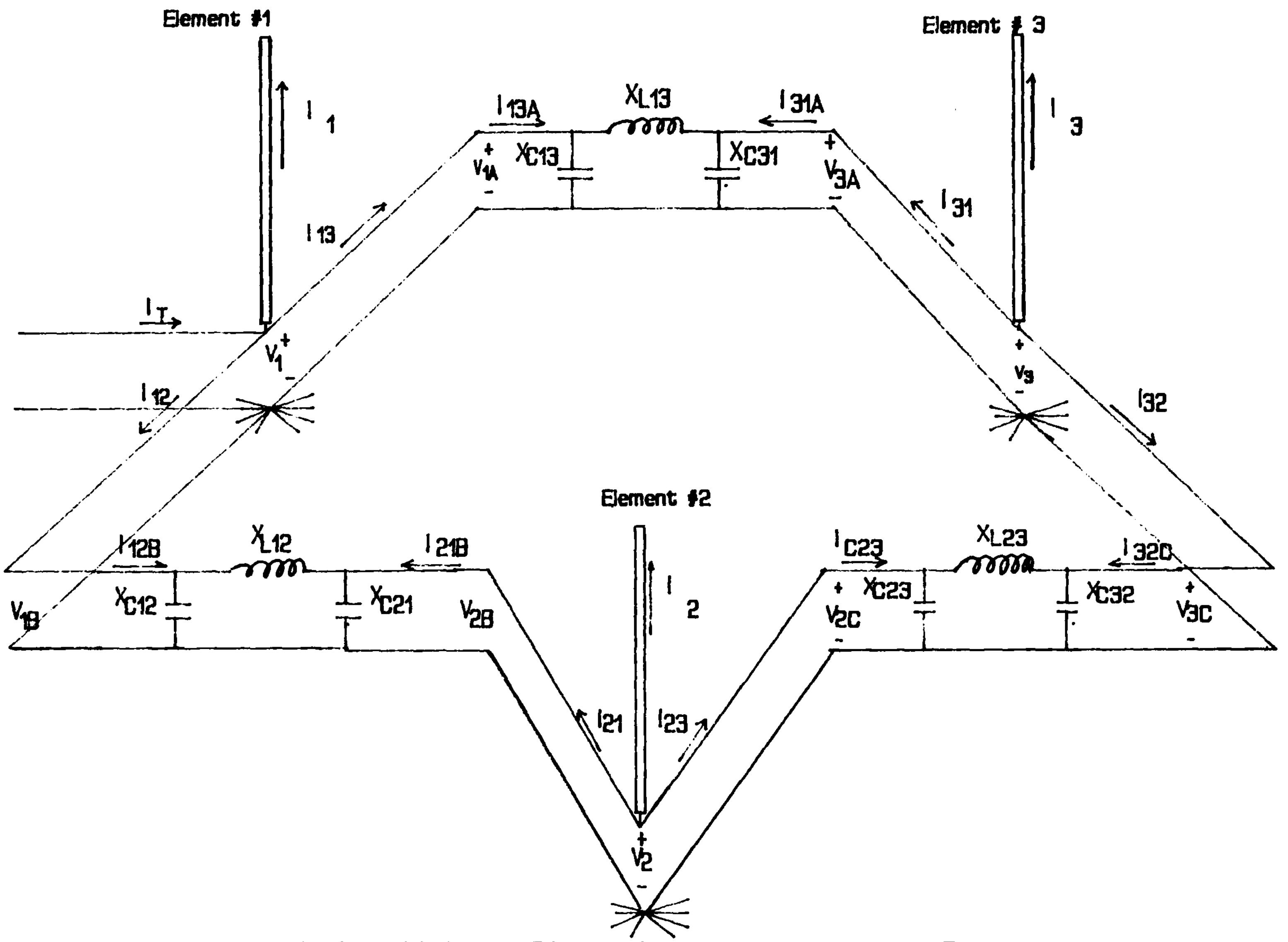


FIGURE 5 - Unique Phased Array Antenna System

elements and the proposed interconnection medium which together define the phased array antenna system to be analyzed and designed in this thesis.

Two independent design methods for determining the parameters of the matching networks have been developed. The first method developed is based on solving a set of nonlinear design equations by an extension of Newton's root-finding method. The word "design" as used in this context means that <u>the current distribution on the elements are given, while the parameters of the PI networks are the unknown values that need to be calculated.</u> Since the design equations are nonlinear, a Taylor's series expansion procedure is used to linearize the set of complex equations. The Taylor's series expansion is performed about an initial guess for the nonlinear variables and the linearized equations are then solved using matrix inversion techniques. A new approximation to the desired solution results from the first iteration of this process; this new approximation is then used as the starting point for the next iteration. This process will be referred to from now on as the iterative matrix solution method.

The procedure described above is similar to Newton's Method for finding a root of f(x)=0, where an initial guess is used as a starting point for what one hopes will be a convergent sequence approaching reasonable values for the root in question.[4] A graphical representation of Newton's iterative root-finding procedure is shown in Figure 6. If the iterative matrix solution method converges, it does so very quickly, as is the case with Newton's Method. Newton's Method can diverge if the initial guess is not fairly close to an acceptable solution and this is also true with the iterative matrix solver routine. Considering the system of equations to be evaluated Figure 6, is quite misleading; it misrepresents the complexity of the equations actually being dealt with in this thesis. The 24 complex equations that describe the phased array system of this thesis are presented in Chapter Four. Consequently, extensive use must be made of large-scale computational techniques and resources.

The second design method presented in this thesis is based upon the quantitative "analysis" of the phased array system of Figure 5 <u>with the reactive parameters of the PI networks being given, while the current distributions on the array become the unknowns to be calculated.</u> Since the nonlinearity in the first design method was caused by the parameters of the PI networks being unknown, the set of analysis equations are linear and the current distributions can be solved for directly (again, because of the number of equations being dealt with, computer implemented matrix methods are used).

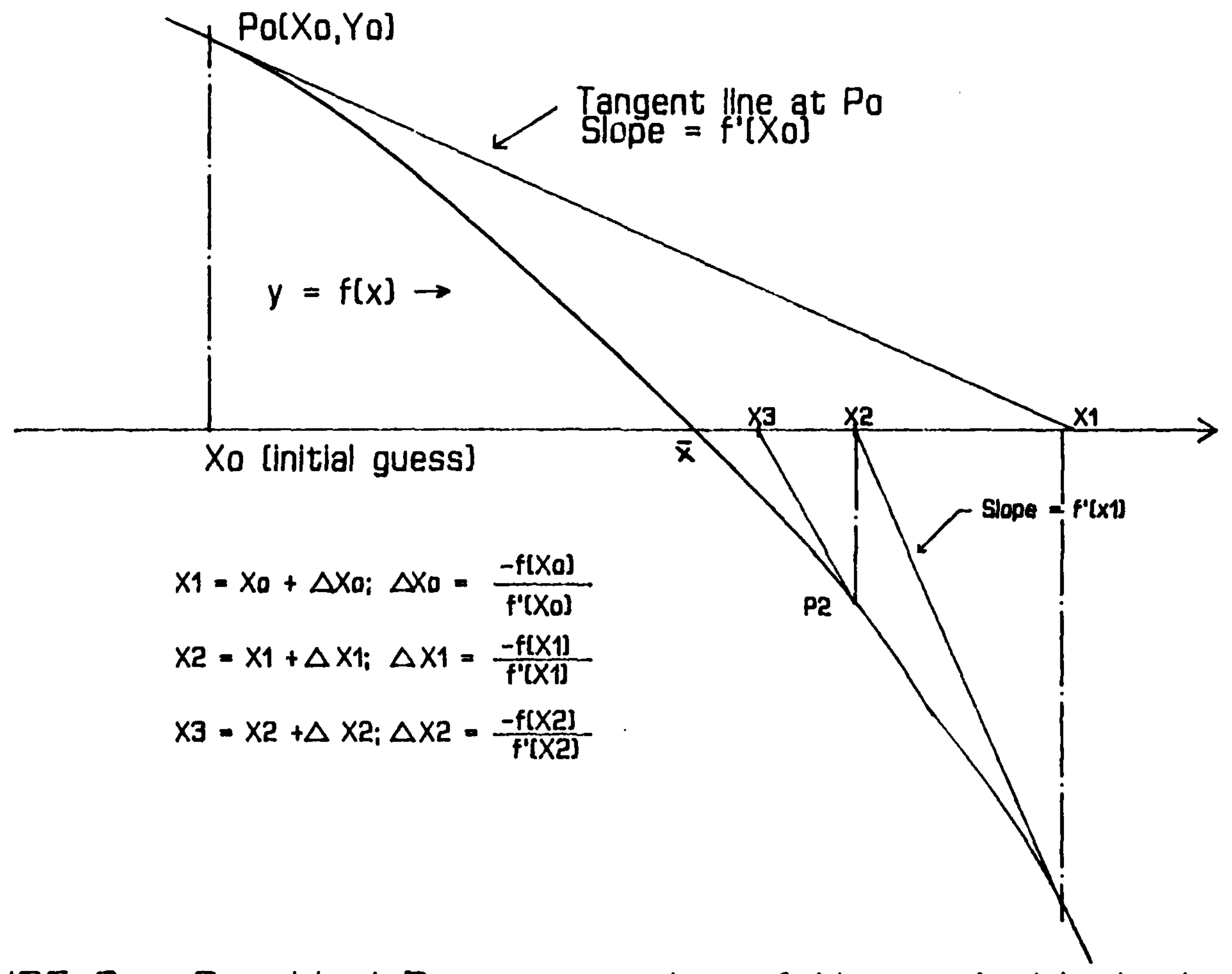

Po(Xo,Yo)
Tangent line at Po
Slope = f'(Xo)
y = f(x) →
Xo (initial guess)
x̄
X3
X2
X1
Slope = f'(x1)
P2
X1 = Xo + ΔXo; ΔXo = −f(Xo) / f'(Xo)
X2 = X1 + Δ X1; Δ X1 = −f(X1) / f'(X1)
X3 = X2 +Δ X2; ΔX2 = −f(X2) / f'(X2)

FIGURE 6 - Graphical Representation of Newton's Method

However, since a particular radiation pattern corresponds to a specific set of relative current distributions, a method for finding the parameters of the PI networks that will in fact produce the desired current distributions must be developed.

A direct search optimization procedure which adjusts the value of the parameters of the PI networks until the element currents are as close as possible to their desired values was developed as a solution to this problem. Initial guesses for the PI network parameters are used to solve the system of complex linear equations describing the equilateral triangular array. Three of the unknown complex variables in this set of equations are the current distributions for each element. The difference between these current distributions and the given (desired) current distributions becomes the criterion upon which the optimization procedure bases its decisions. This difference is then squared and is designated the hill height magnitude. The procedure is a type of feedback enforced optimization routine where the present values of the current distribution on the antenna elements are continually fed back to the decision module of the optimization routine in order for the PI network parameters to be adjusted accordingly.

The unique direct search optimization method used in the constrained "analysis" design method of Chapter

Seven was developed in this thesis as a possible tool for solving the nonlinear and linear equations describing the equilateral triangular array. It was developed out of discussions with the author's thesis advisor concerning multivariable optimization techniques.[5] This routine will adjust the value of the variables of any function dependent upon the magnitude of the function. Since the function's present value is based on the current values (within the procedure) of the variables, these variables will progress in whatever direction will lower the magnitude of the function. The routine is adaptive; it has the capability of increasing and decreasing the step size of the variables. This gives the routine the power to move the variables in small or large increments. Also, this routine is able to detect when a variable is close to a possible solution. A variable is close to a possible solution when the routines decision module begins to periodically adjust the variable's magnitude above and below the particular magnitude that has (up to this point in the process) resulted in the best solution. When this oscillatory behavior is detected, the magnitude of the changes in the variable's step size is reduced.

Early development of the first design method was performed in APL on a IBM PC computer, but once the computation burden of the iterative matrix solver problem went beyond the capabilities of the IBM PC, developement of the procedure was moved to FORTRAN code on a VAX 11/780 computer system. The second design method was completely developed in FORTRAN on a VAX 8600 cluster. The linear equation solver LEQ2C in the standard library subroutines from the IMSL scientific subroutine package was used to solve the matrix equations of the analysis.

In Chapter Two, the basic principles of radiation patterns are discussed. The particular characteristics used to determine the radiation pattern in a phased array are listed and reviewed. The interaction of the radiating fields from each element in the array is demonstrated and a procedure for its calculation is presented. Chapter Two is completed with an in depth evaluation of a method of beam steering which allows a user to specify directions of maximum and minimum radiation.

The methods using reactive networks for impedance matching an antenna driving point impedance to the characteristic impedance of a transmission line are discussed in Chapter Three. The numerous ways of impedance matching are listed, as well the factors used in deciding which method is best suited for the particular application. Specific examples using PI and TEE impedance matching networks are presented with calculations. A two element vertical phased array is matched using PI networks in an extensive presentation of the calculations involved in the analysis of this simple array.

In Chapter Four, a detailed description of the physical and electrical characteristics of the proposed phased array system of Figure 5 will be presented. The equations that characterize the transmission lines, the PI networks, the antenna mutual impedance relations, and Kirchoff's Current Law are reviewed. The quantitative analysis of these 24 equations, which result in a solution for the currents on the radiating elements, is performed at the end of Chapter Four.

Presentation of a interconnection network design procedure, using an extension of Newton's Method for finding the roots of a quadratic equation, will be demonstrated in Chapter Five. Developement of the procedure entails linearizing a set of 14 nonlinear design equations using a Taylor's series expansion about the initial guesses for some of the complex unknown variables. This linearization process is shown in detail and the resulting linear equations are listed. This chapter concludes with example designs and the patterns produced by those designs.

Chapter Six is an in-depth discussion of the unique direct search optimization method developed by the author as part of this thesis. The fundamental decision control procedures are explained using a quadratic function of two variables as an example. This method has been developed into a n-variable optimization routine in the form of a FORTRAN subroutine. Example functions are shown along with the results generated in attempting to minimize the magnitude of their hill heights.

The direct search optimization routine is utilized in Chapter Seven where it is applied to a constrained "analysis" design technique. The design technique of Chapter Seven uses the analysis equations that describe the equilateral phased array to try and enforce a desired current distribution on the array elements. Since solving the analysis equations results in a current distributions on the elements of the array, this design technique will use the optimization routine of Chapter Six to adjust the choosen PI network parameters until the desired current distribution is realized. Sample runs of this design technique and the resulting radiation patterns are presented and discussed.

# II. BEAM STEERING IN PHASED ARRAY ANTENNA SYSTEMS

The electrical field produced by an array of radiating elements can be determined by vector addition of the fields produced by each individual element. Phased array antenna systems produce a pattern with directivity when the fields from the elements interfere constructively in the direction of interest and cancel each other out in the remaining areas (null directions). From a theoretical and computational standpoint, this can be done very easily. However, from a practical point of view, the potential performance of the phased array can only be approached. There are four items in a multi-element phased array that determine the radiation pattern of the array as a whole. These items are:[6]

1. The relative displacements between each of the individual elements.
2. The amplitude of the current flowing into each element.
3. The phase of the current flowing into each element.
4. The radiation patterns of each individual elements.

In this thesis, items 1 and 4 are assumed fixed, and attention is focused entirely on the production of feed currents with appropriate amplitudes and phases.

The method for radiation pattern control used in this thesis allows a user to manually input the pattern desired by specifying directions of maximum radiation ($\theta_m$) and minimum radiation ($\theta_n$), and receive as output the complex currents required on each element to produce the desired pattern. Of course, even if the proper current distribution is on the array, the prospects for production of exactly the desired pattern are marginal at best. The inaccuracy of the self and mutual impedance values are caused by a non-infinite ground plane. The inaccuracies associated with the measurement of the elements' input current also contributes to the degradation of the radiation pattern.

When evaluating radiation patterns of vertical phased arrays, one considers the radiating elements to be isotropic point sources. Looking down upon a single element radiating in free space one sees an omnidirectional pattern in the horizontal (azimuthal) plane. The field intensity in any direction from the element can be represented by a vector $R\underline{/\theta}$ , where R is the magnitude of the field in the direction $\theta$ (see Figure 7). The angle $\theta$ is defined as the angle the vector R makes with the positive X axis in a counterclockwise direction.

When two or more vertical elements are configured in a geometrical pattern or array and powered from a

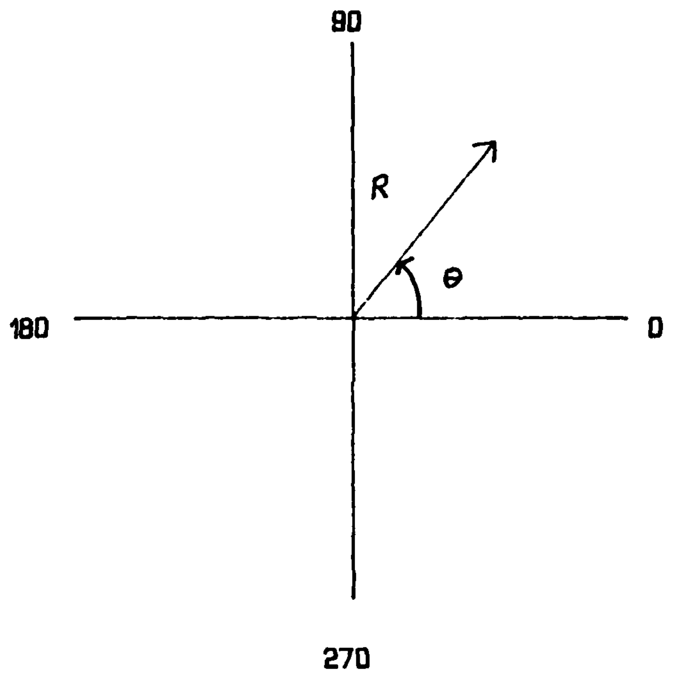


FIGURE 7 - EM Field Intensity

common source, the combination of each element's radiation results in more radiation in some directions (constructive interference) and less in others (destructive interference). A radiation lobe results from constructive interference while a null is caused by destructive interference. Also the radiation pattern is directly dependent on the time delay resulting from the physical separation between the elements of an array. It is this time delay which causes the phase shifts and leads to constructive/destructive interference.

This time delay is also referred to as a geometric phase shift and an equation defining this effect can be derived using some geometric principles. Figure 8 is a representative drawing of two radiating elements in the XY plane and from it one can observe quite clearly the phase shift from a infinite observation point. From Figure 8 it can b[illegible]hown that the angle equals

$$\phi = \mathrm{Tan}^{-1} \frac{Yi}{Xi} \qquad (2.1)$$

where

$$\psi = \phi - \theta$$

or

$$\psi = \mathrm{Tan}^{-1} \frac{Yi}{Xi} - \Theta \qquad (2.2)$$

From both elements in Figure 8 there extends dashed lines which represent the line of observation from infinity to each of the elements. Through the element #2 located at $(X_i, Y_i)$ and the line of observation from element #1 there is an imaginary plane of reference which is perpendicular to both lines of observation. This plane of reference is used to determine the electrical distance between it and element #1. This distance represents a time delay and indicates that the EM radiation from element #1 is behind or lagging the EM radiation from element #2. The distance can be defined as

$$X = R\mathrm{Cos}\,\psi$$

where

$$R = \sqrt{X_i^{\,2} + Y_i^{\,2}} \qquad (2.3)$$

and

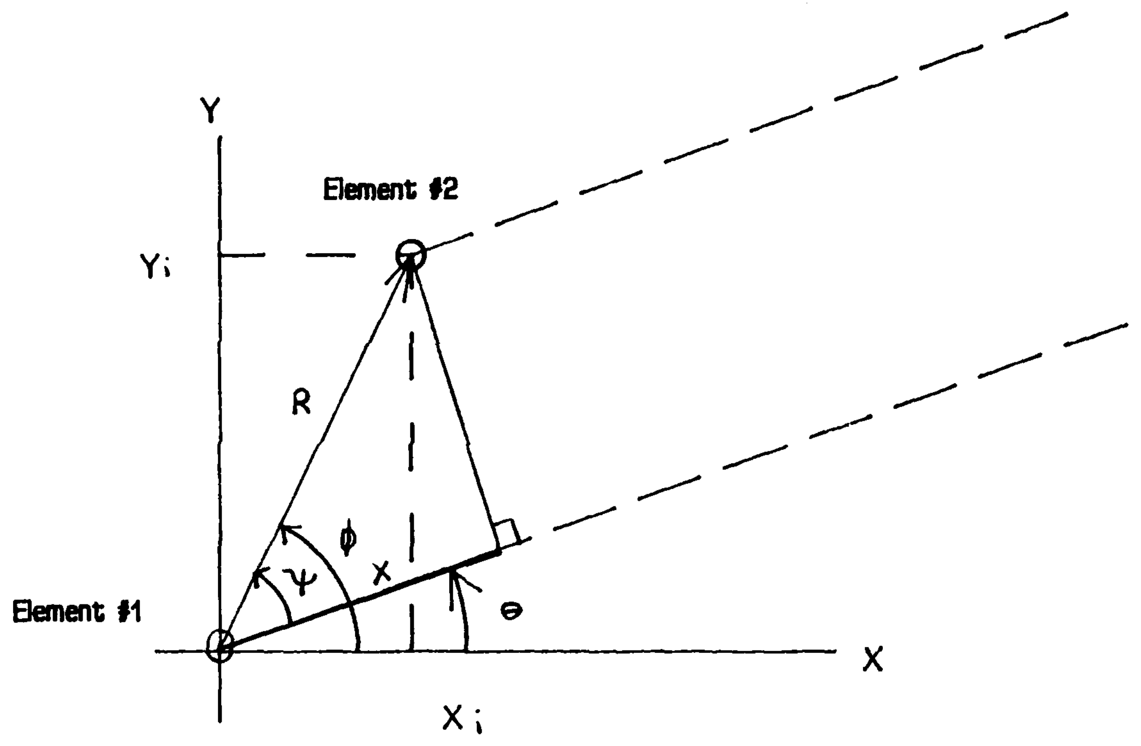


FIGURE 8 - Geometric Phase Shift

2. Provided the necessary phase shift.
3. Provide the desired current.
4. Provide the desired power.

$R_1$ and $R_2$ are the input impedance to the PI network and they are purely resistive. The desired phase shifts across the two networks are $B_1$ and $B_2$; $I_1$ and $I_2$ are the driving point currents at the base of each antenna, and $Z_{L1}$ and $Z_{L2}$ are the driving point impedances. The driving point impedances for this two element array can be calculated using the mutual impedance equations characteristic of this array.[19] The mutual impedances equations for this array are

$$V_1=I_1Z_{11} + I_2Z_{12} \tag{3.7}$$

$$V_2=I_1Z_{22} + I_2Z_{21} \tag{3.8}$$

With quarter-wave spacing of quarter-wavelength verticals over an infinite infinitely conducting ground plane, the self-impedances ($Z_{11}$,$Z_{22}$) and the mutual impedances ($Z_{12}$,$Z_{21}$) have the following complex values,[20]

$$Z_{11} = Z_{22} = (36.5 + j21)$$

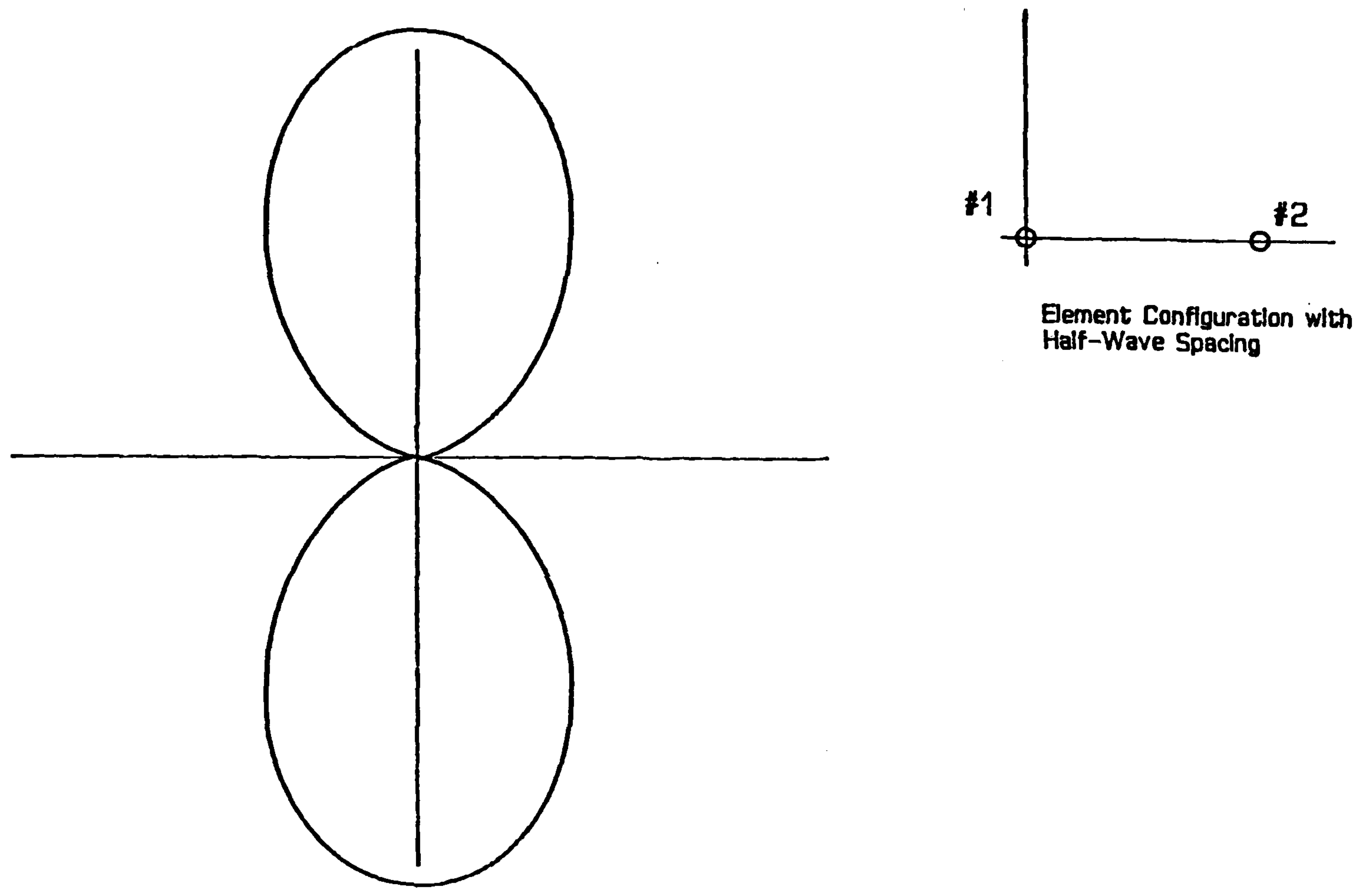


FIGURE 9 - Broadside Radiation Pattern

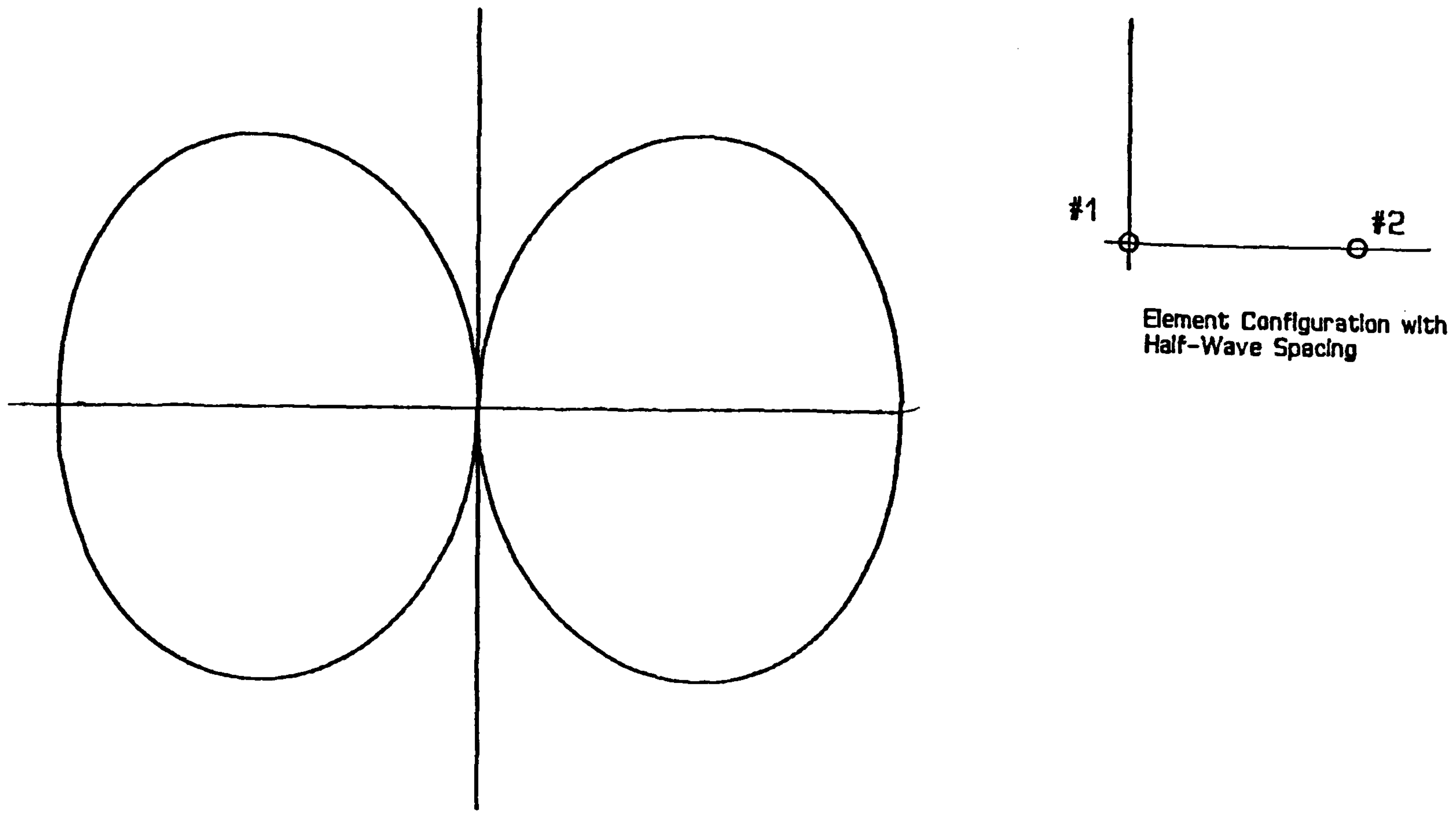


FIGURE 10 - Endfire Radiation Pattern

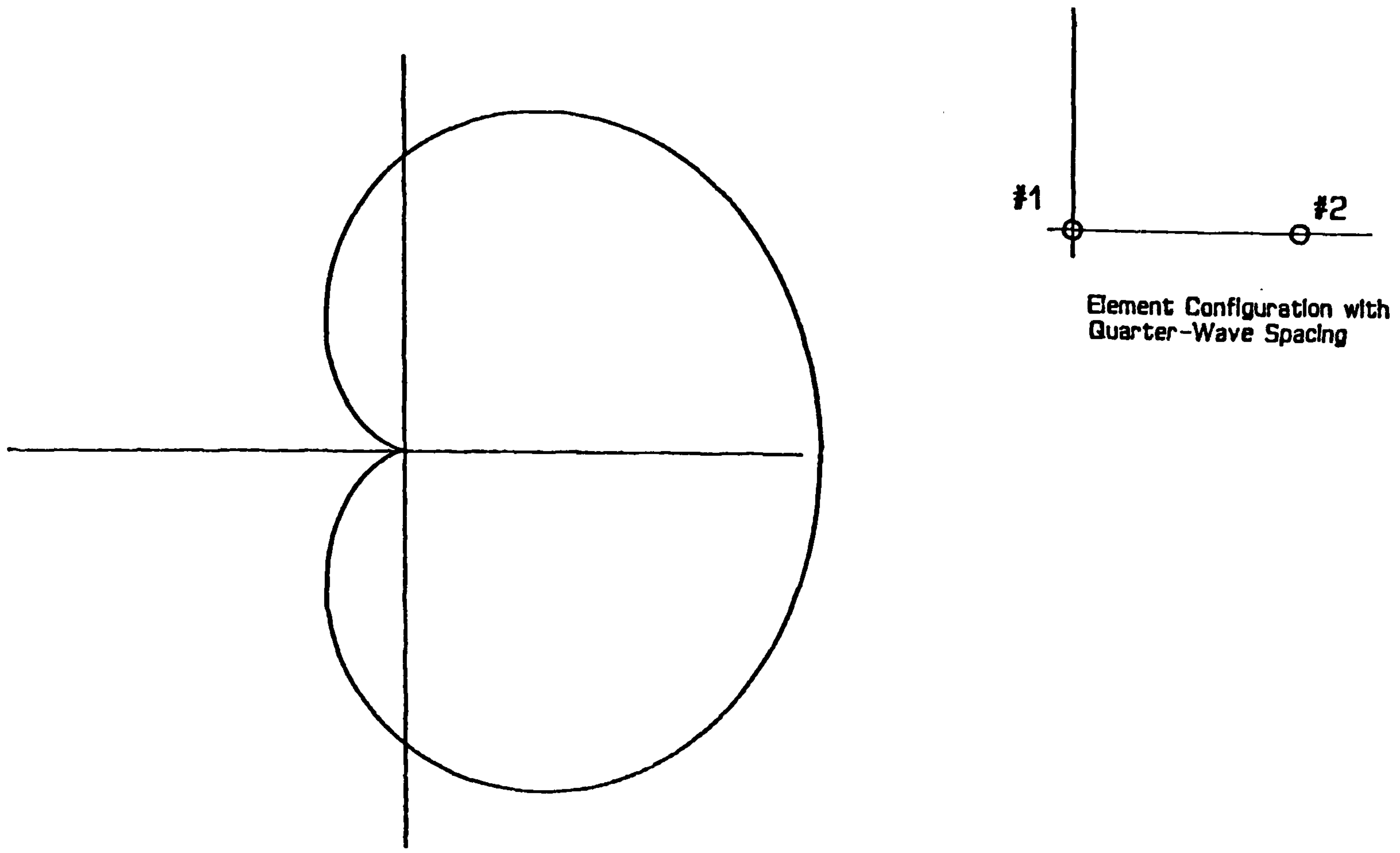


FIGURE 11 - Cardiod Radiation Pattern

located at points in a horizontal plane defined in XY coordinates. The field intensity F at a distant point in the direction $\theta$ is the sum of fields $E_i$ generated by each element,[10]

$$F = \left| \sum_{i=1}^{N} E_i \right| \tag{2.5}$$

where N is defined to be the total number of radiating elements in the array, and $E_i$ is given by

$$E_i = \sqrt{(E_{xi})^2 + (E_{yi})^2} \tag{2.6}$$

where

$$E_x = \sum_{i=1}^{N} A_i \operatorname{Cos}(\theta_e + \theta_g)$$

$$E_y = \sum_{i=1}^{N} A_i \operatorname{Sin}(\theta_e + \theta_g)$$

with

$\theta_e$ = electrical phase

$\theta_g$ = geometric phase

The amplitude of the ith element's current is $A_i$ and $\theta_e$ is the phase of the ith element's input current. $\theta_g$ is the geometric phase shift caused by the elements location in the xy plane and is defined by equation (2.4).

The magnitude of the field intensity is based in part on the location of the elements but more importantly on the angle of observation. Therefore the radiation pattern at any point around the array can be computed by calculating the total field magnitude F as a function of (the observation angle). The calculation implied by equation 2.5 is easily implemented in FORTRAN; a computer program named HILPAT will calculate and plot the radiation pattern of N vertical antennas spaced at any desired locations in the XY plane with arbitrary excitation currents is given in Appendix A. This program assumes the antenna placed at the origin (0,0) in the XY plane is the phase reference ($\theta_{e1}$ equals 0 degrees).

By definition, a null in a radiation pattern is a direction in which the field intensity equals zero.[11] Thus a null in the direction $\theta_n$ occurs when the net field intensity F satisfies

$$\left| \sum_{i=1}^{N} E_i \right|_{\theta = \theta_n} = 0 \qquad (2.8)$$

Since each $E_i$ is a complex quantity, the above equation implies that

$$\sum_{i=1}^{N} A_i \operatorname{Cos}( \theta_e + \theta_g ) = 0 \tag{2.9}$$

and

$$\sum_{i=1}^{N} A_i \operatorname{Sin}( \theta_e + \theta_g ) = 0 \tag{2.10}$$

Therefore a single null in direction $\theta_n$ can be produced if equations 2.9 and 2.10 are satisfied.[12] The 2N unknowns ($A_i$, i=1 to N and $\theta_{ei}$, i = 1 to N) in this nonlinear transcendental set of equations can make solving them quite difficult.

Solution of equations (2.9, 2.10) can be greatly simplified if the phases $\theta_{ei}$ are known, for then (2.9) and (2.10) become two linear equations in the variables $A_i$. For an N antenna system with a given set of phases $\theta_{ei}$, N/2 null directions can be produced in the radiation pattern by solving N linear equations for the current amplitudes $A_i$.[13]

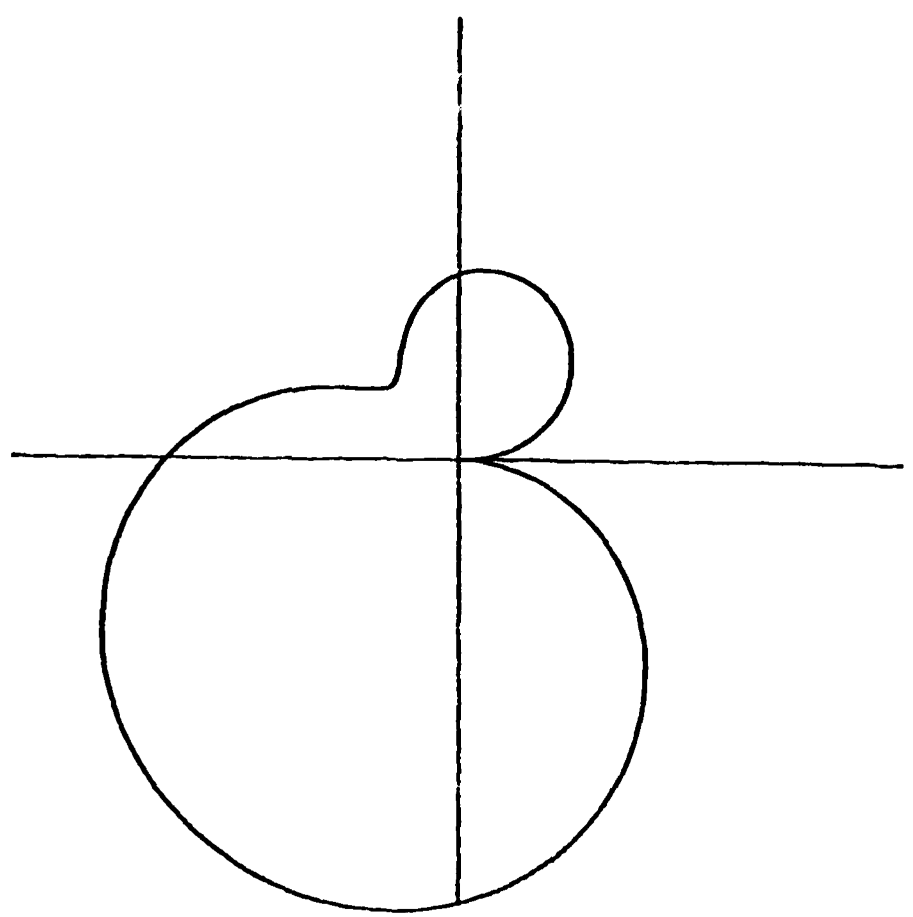
Since one is also interested in the specific direction of the "main lobe" of the pattern (particularly when the antenna is being used for transmission), it also will be necessary to find a set of phases that will produce as much radiation as possible in the desired direction. The current amplitudes $A_i$ are constrained by the linear equations describing the null direction and can not be directly adjusted to provide the desired "main lobe" direction.[14] However, the $A_i$'s depend on the $\partial_{ei}$'s, which allows some measure of indirect control over the main lobe.

The method for maximizing the radiation pattern in a specific direction $\theta_m$ uses a direct search optimization routine. This optimization routine was used to adjust phases $\theta_{ei}$ to whatever values would provide maximum radiation in $\theta_m$ under the constraints imposed by the specified $\theta_n$. Starting values for the phases $\theta_{ei}$ are input by the user and after a specified number of iteration the phases that produce maximum radiation, in conjunction with the current amplitudes defined by $\theta_n$, will be output and may be used in the plotting of the resulting radiation pattern.

The described beam steering procedure was adapted to the three element equilaterally spaced triangular array in Figure 1. A computer program (NULL) was developed to solve equations (2.9) and (2.10) for the $A_i$ (Appendix A) and a direct search optimization routine (MAX_HC) was utilized in the constrained amplitude, phase maximization procedure described in the previous paragraph. Since there are only three (N = 3) elements in the equilateral triangular array, only one null direction can be specified along with a direction of maximum radiation. This direction of maximum radiation is of interest to a user for both transmitting and receiving purposes. Knowing the direction of maximum signal strength allows one to determine the direction of the signal being sent out or received.

The program developed for determining radiation patterns for the 3 element array of Figure 1 was run with varying $\theta_m$ and $\theta_n$ inputs and some resulting patterns are shown in Figures 12 and 13. The main program MAX_RAD uses the subroutines NULL and MAX_HC to generate those resulting radiation patterns an are listed in Appendix A. The specified null directions must be kept a reasonable number of degrees away from the desired main lobe in order for acceptable patterns to result. Null and main lobe seperation angles of greater than 25 degrees provide patterns having the best main lobe maximization results, with the specified null always being present (because the null locations are treated as constraints that the direct search optimization procedure has no choice in adjusting).

The current amplitudes and phases resulting from this beam steering method are used as input for both of the design procedures developed in this thesis. In the design by linearization procedure, the current amplitudes and phases for a particular pattern are used as the given current distributions for the array elements with the solution of the nonlinear equations resulting in values



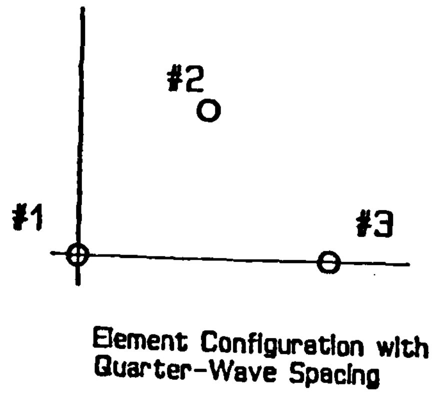
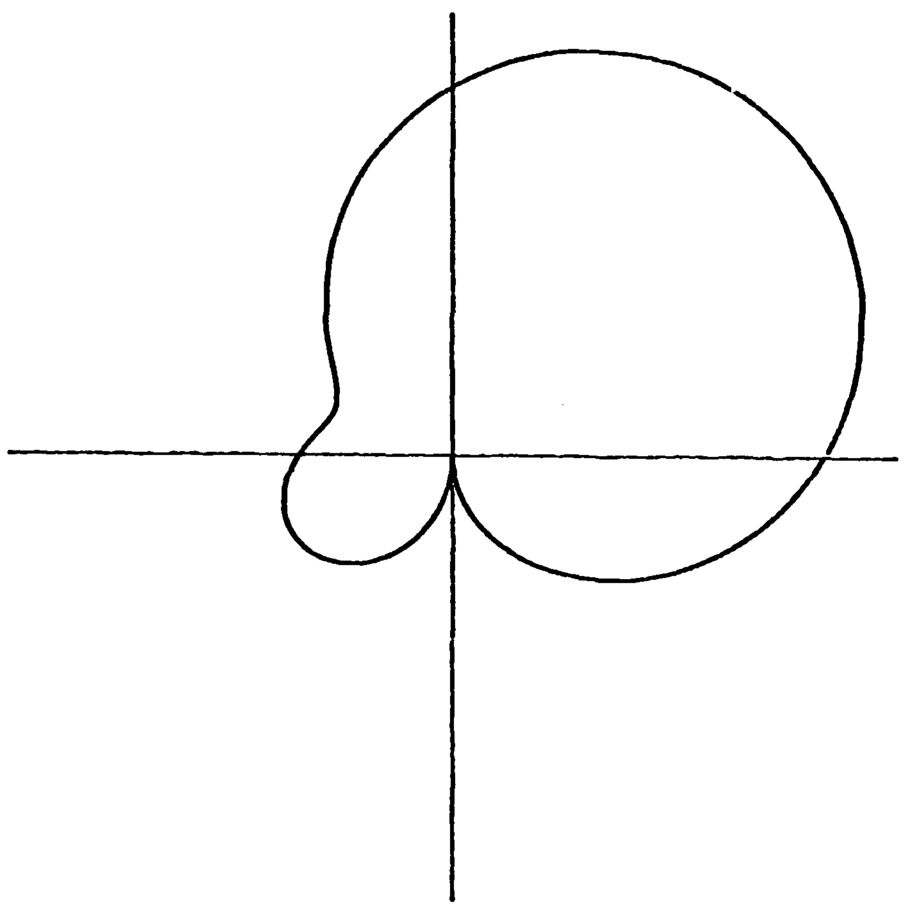


FIGURE 13 – 3 Element Radiation Pattern
$\Theta m = 240$ and $\Theta n = 0$



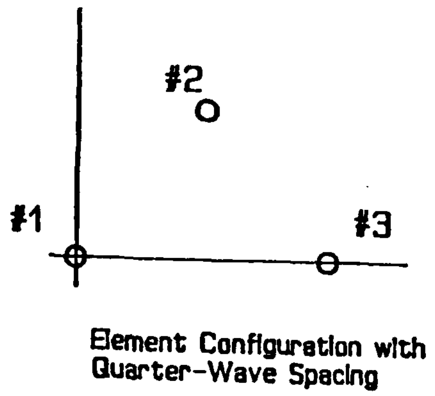


FIGURE 12 - 3 Element Radiation Pattern

$\theta m = 60$ and $\theta n = 270$

for the unknown PI network parameters. The current distributions are used in the constrained "analysis" design procedure of Chapter Seven as the given (desired) solution to the linear version of the equations describing the test array. Since the PI network parameters are the controlling components in this design method, they will be adjusted by the direct search optimization routine of Chapter Six until the desired current distributions are realized.

The beam steering procedure or the means of determining the current distribution for a desired radiation pattern is of primary concern during the early phases of designing a phased array antenna system. Networks that will in fact produce the desired pattern are in themselves a separate topic the designer must consider once the current distributions for the array elements are known. An approach to designing the phasing networks is the primary concern of this thesis. The beam steering procedure described above was simply adapted to the author's three element triangular array in Figure 1.

This chapter has provided the reader with a brief introduction into the theory behind the radiation patterns generated in phased array antenna systems. The following topics were discussed: the four principle determinants of a phased array's radiation pattern, the derivation of the equation describing the geometric phase shift, and the use of the direct search optimization routine to maximize the radiation in a specified direction ( $\Theta_m$) while maintaining the constrained direction of minimum radiation ( $\Theta_n$). The programs used to implement the calculations and plotting of radiation patterns for the equilateral array were also discussed.

# III. FEED NETWORKS FOR FIXED PATTERN PHASED ARRAYS: PRIOR ART

When designing an antenna system, in addition to calculating the current distributions corresponding to the desired radiation pattern, one must construct a network or system of networks that supplies these desired current amplitudes and phases to each element. Traditionally this is done using impedance matching networks connected to the base of each antenna. These networks produce a desired phase lag or lead to each particular antenna while also matching the driving point impedance of the antenna to the characteristic impedance of the transmission line.

There are many different methods one could use to match the antenna driving point impedance to the transmission line supplying the current. The decision to use a particular method depends on many factors. Some of the factors that need consideration are: the element operating currents and voltages, the frequency of operation, the degree of initial impedance mismatch, the equipment available for measurement and construction of the system, the required bandwidth, the available area for operation and construction, and economics. The most commonly used methods are:[15]

1. Design of load and feeder to have equal impedance.
2. Use of coupling reactance networks.
3. Use of tapped transmission lines.
4. Using a series section of transmission line as an impedance transformer.
5. The use of a stub section of transmission line as a reactance in parallel with the power source at a point where the impedance at that particular point will be equal to the lines charactersitic impedance.
6. The use of a reactance component in place of a stub line and electrically equivalent to it.
7. The use of a coupled section of line in parallel with the power source with the length needed to reflect the correct amount of reactance into the main power source at the point where an impedance match would occur.
8. The use of a tapered transmission line as a broadband impedance matching transformer.

A vertical antenna of fixed frequency is used most often at broadcast radio stations and is usually matched using a coupling network of reactances in series with the coaxial transmission line from the transmitter. Also, an amateur radio operator interested in only one frequency of operation would probably find this method of impedance matching most practical.

## A. Impedance Matching Networks

With this in mind, an example of an impedance matching network designed to match the 50-ohm coaxial transmission line from a transmitter to an antenna having a purely resistive impedance of 100 + j0 ohms is shown below. A phase difference of +30 degrees, between the transmitter current and the current flowing through the element is incorporated into this design. Therefore, the phase at the output lags the phase of the input by 30 degrees. A PI network, as shown in Figure 14, will be used as an example.

Since we are assuming the network is ideally reactive with zero losses the components of the network $Z_A$, $Z_B$, and $Z_C$ can be directly computed from the formulas below:[16]

$$Z_A = j \frac{R_1 R_2 \text{ Sin } B}{R_2 \text{ Cos } B - R_1 R_2} \tag{3.1}$$

$$Z_B = j \frac{R_1 R_2 \text{ Sin } B}{R_1 \text{ Cos } B - R_1 R_2} \tag{3.2}$$

$$Z_C = j R_1 R_2 \text{ Sin } B \tag{3.3}$$

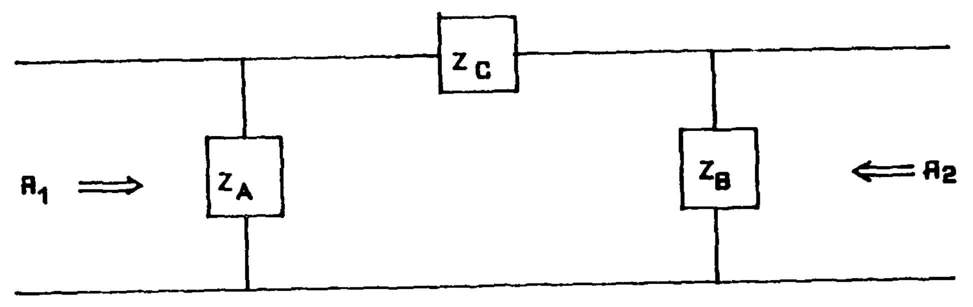


FIGURE 14 - General Diagram of a PI Network

B is the phase difference between the input and output terminals of the PI networks, while $R_1$ and $R_2$ are the image impedances with $R_1 = R_{in}$ and $R_2 = R_L$. The driving point impedance is a more specific term for the image impedance and it is defined as the impedance at the base of a radiating element (vertical antenna). For our example,

$$Z_A = j\frac{100*50\ \mathrm{Sin}(30)}{50*\mathrm{Cos}(30) - 100*50} = j157.31$$

$$Z_B = j\frac{100*50\ \mathrm{Sin}(30.0)}{50*\mathrm{Cos}(30.0) - 100*50} = -j91.21$$

$$Z_C = j\ 100*50\ \mathrm{Sin}(30.0) = j35.36$$

and the resulting network is shown in Figure 15.

If a TEE network as shown in Figure 16 is used in place of the PI network, the following equations would apply:[17]

$$Z_1 = -j\ \frac{R_1\ \mathrm{CosB} - R_1\ R_2}{\mathrm{SinB}} \qquad (3.4)$$

$$Z_2 = -j\ \frac{R_1\ \mathrm{CosB} - R_1\ R_2}{\mathrm{SinB}} \qquad (3.5)$$

$$Z_3 = -j\ \frac{R_1\ R_2}{\mathrm{SinB}} \qquad (3.6)$$

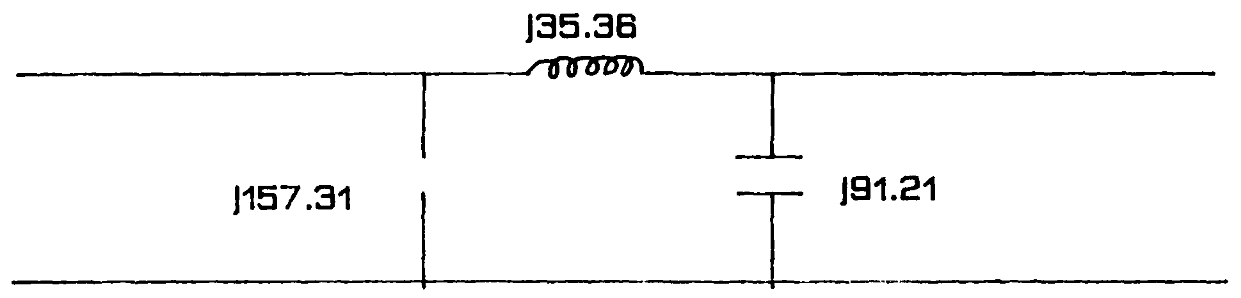


FIGURE 15 - Resulting PI Network for R1=50 R2=100

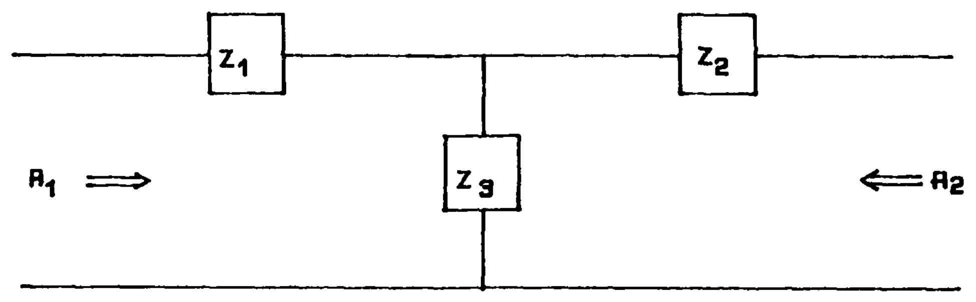


FIGURE 16 - General Diagram of a TEE Network

and the resulting reactances being

$$Z_1 = j54.82$$
$$Z_2 = -j31.78$$
$$Z_3 = -j141.42$$

with the resulting network shown in Figure 17.

Usually in the design of PI or TEE matching networks the desired input and output impedances as well as the phase shift are known and the network components are the unknowns.

Another design example will be presented with the desired input and driving impedance being given as $Z_L = 75 - j30$ and $Z_{in} = 600 + j150$. The phase shift of the network will be the difference between the phase of the input current of the network and the phase of the base current of the antenna of a positive 60 degrees (phase lead). Since the B in equations (3.1-3.6) represents phase lag, a negative B will be used in the calculations in order to keep the signs of the network components correct. For the PI network of figure 14 the following values were calculated using equations 3.1, 3.2, and 3.3 after transforming the complex series input and load impedances to their parallel equivalent circiuts.

$$Z_A = +j155 \qquad \text{(a 155-ohm inductor)}$$

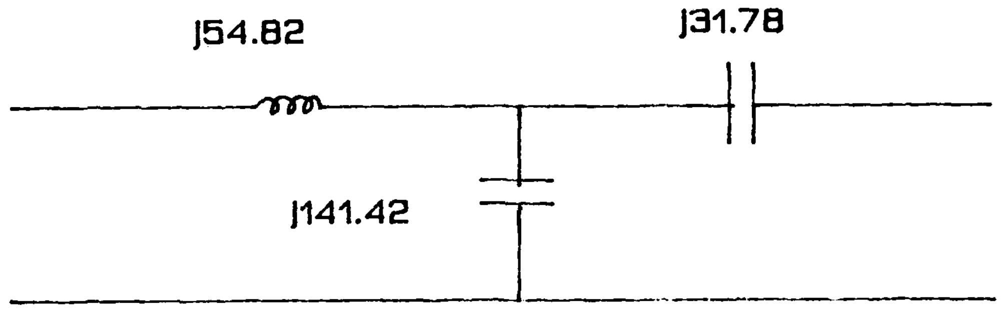


Figure 17 - Resulting TEE Network for R1=50 and R2=100

$Z_B = -j113$ (a 133-ohm capacitor)

$Z_C = -j108$ (a 108-ohm capacitor)

and the resulting network is shown in Figure 18 (remebering that the required network is lossless).

Application of the PI network to a specific phased array antenna system will show how these networks are used by an array designer to provide efficient impedance matching with the desired phase shift in all transmission lines leading from the transmitter to the array elements.

## B. Conventional Phased Array Design

A two element vertical array with impedance matching networks is shown in Figure 19. The following design approach is most often used by radio broadcast stations and amateur radio operators in the LF to HF frequency ranges using vertical antennas positioned above the surface of the earth (non-infinite ground plane). The transmitter will supply power to both elements and therefore a PI network will be placed in series with the transmission lines leading from the transmitter to the antennas.[18] The PI networks must accomplish four things:

1. Provide an impedance match between transmitter and the antenna.

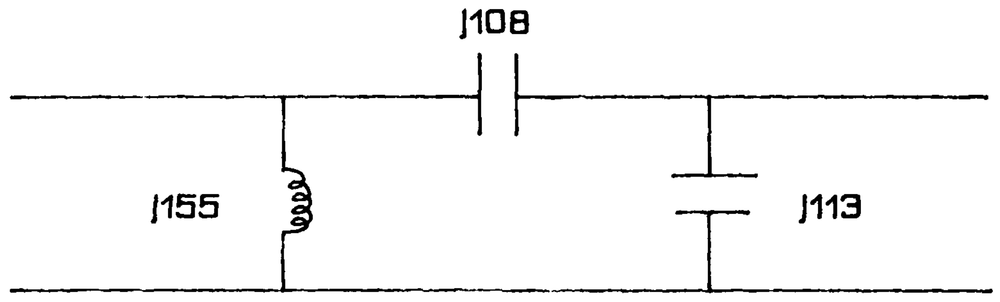


Figure 18 - PI Network Designed from Complex Impedances

$$X = \sqrt{X_i^2 + Y_i^2} \quad \text{Cos}\left[ \text{Tan}^{-1}\left(\frac{Y_i}{Xi}\right) - \theta \right] \qquad (2.4)$$

The distance X is expressed in wavelengths and can be converted to radians by multiplying by $2\pi/\lambda$.

Classical examples of basic radiation patterns producible by such simple arrays are the endfire and broadside configurations. The broadside configuration consists of two vertical antennas fed in phase yielding a radiation pattern as shown in Figure 9.[7] The endfire array also has two verticals (with the same spacing), but they are fed 180.0 degrees out of phase. The radiation pattern produced by this arrangement is shown in Figure 10.[8]

The endfire array can be varied in spacing and phase in such a way as to produce unidirectional patterns. When fed 90 degrees out of phase the direction of maximum radiation is always in line with the two verticals towards the direction of the antenna receiving the lagging excitation. A classical example of this phenomena is the cardiod pattern shown in Figure 11. This pattern was produced with two quarter wave verticals spaced a quarter wave apart and excited 90 degrees out of phase.[9]

We now proceed to analyze a vertical phased array consisting of N antennas. The vertical elements are

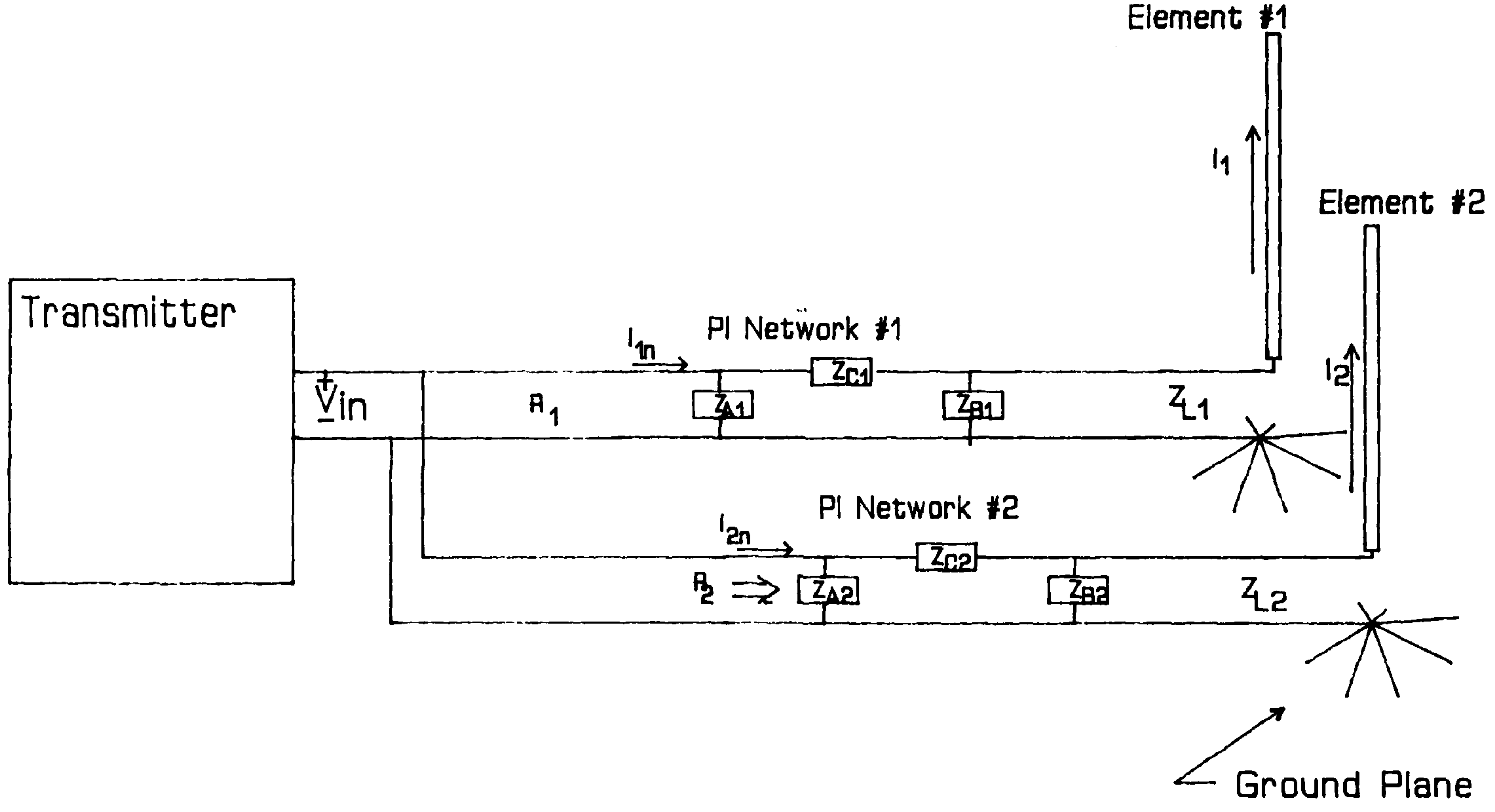


FIGURE 19 - Two Element Feed Network

$$Z_{12} = Z_{21} = (20.4 - j14.2)$$

The driving point impedances of each antenna are

$$Z_{L1} = \frac{V_1}{I_1} \qquad (3.9)$$

$$Z_{L2} = \frac{V_2}{I_2} \qquad (3.10)$$

and the power delivered to each of the elements can be defined in terms of the driving point currents or voltages using

$$P_i = \text{Real}(V_i \cdot I_i^*) \qquad (3.11)$$

or

$$P_i = \left|I_i\right|^2 R_{Li} \qquad (3.12)$$

where (i) defines the particular element in question and RLi is the resistive component of the ith element driving point impedance.[21]

Assuming each PI network will consist of purely reactive components, the input power to the ith network equals the output power ($P_i$) of each network. The total input power to the array can then be expressed as

$$P_T = \sum_{i=1}^{N} P_i \qquad (3.13)$$

Since one would want the input impedance to the array to match the characteristic equation of the transmission line, we have

$$R_{in} = Z_o \qquad (3.14)$$

and by using 50-ohm coaxial line the the expression for the input power becomes

$$P_T = \frac{V_{in}^{2}}{50} \qquad (3.15)$$

with Vin being the voltage from the transmitter.[22]

The input power of the ith network can be used to find the input impedances for each network by defining the phase of the input voltage ($V_{in}$) as 0.0 degrees. Equation 3.15 then takes the following form

$$V_{in} = \sqrt{50 \sum_{i=1}^{N} P_i} \qquad (3.16)$$

or

$$P_i = \text{Real} \frac{|V_{in}|^2}{R_i} \tag{3.17}$$

With the phase of Vin equal to 0.0 degrees, the input resistance to each networks must be

$$R_i = \frac{|V_{in}|^2}{P_i} \tag{3.18}$$

The PI networks must be able to match any complex driving point impedance $Z_{Li}$ at the base of each antenna with any input resistance $R_i$. By using the PI network in this example, one can utilize equations (3.1-3.3) to calculate the components of the networks once the driving point and input impedance that result from a given current excitation have been calculated using equations (3.12) and (3.18). The network must also produce the desired phase shift.

## C. Feed Network Design Example

The design method discussed will be used in the following specific example. In Figure 20 there are two quarter-wave vertical antennas spaced a quarter wave apart above an assumed infinite ground plane. The base currents $I_1$ and $I_2$ are given as desired values and in phasor form they equal

$$I_1 = 1\underline{/0} = 1 \qquad I_2 = 1\underline{/90} = j1$$

With these currents the driving point impedances, using the mutual impedance equations 3.7 and 3.8 and equations 3.9 and 3.10, are found in the following manner;

$$Z_{L1} = Z_{11} + Z_{12}\frac{I_2}{I_1} \tag{3.19}$$

$$= 36+j21+(20.4-j14.18)(j1)$$

$$= (36 + 14.2) + j(21.0 + 20.4)$$

$$Z_{L1} = 50.2 + j41.4$$

$$Z_{L2} = Z_{22}\frac{I_1}{I_2} + Z_{21} \tag{3.20}$$

$$=(36 + j21)(-j1)+(20.4-j14.18)$$

$$= (21.0 + 20.4) - j(36 + 14.18)$$

$$Z_{L2} = 41.4 - j50.2$$

The power for the each antenna is then solved for using

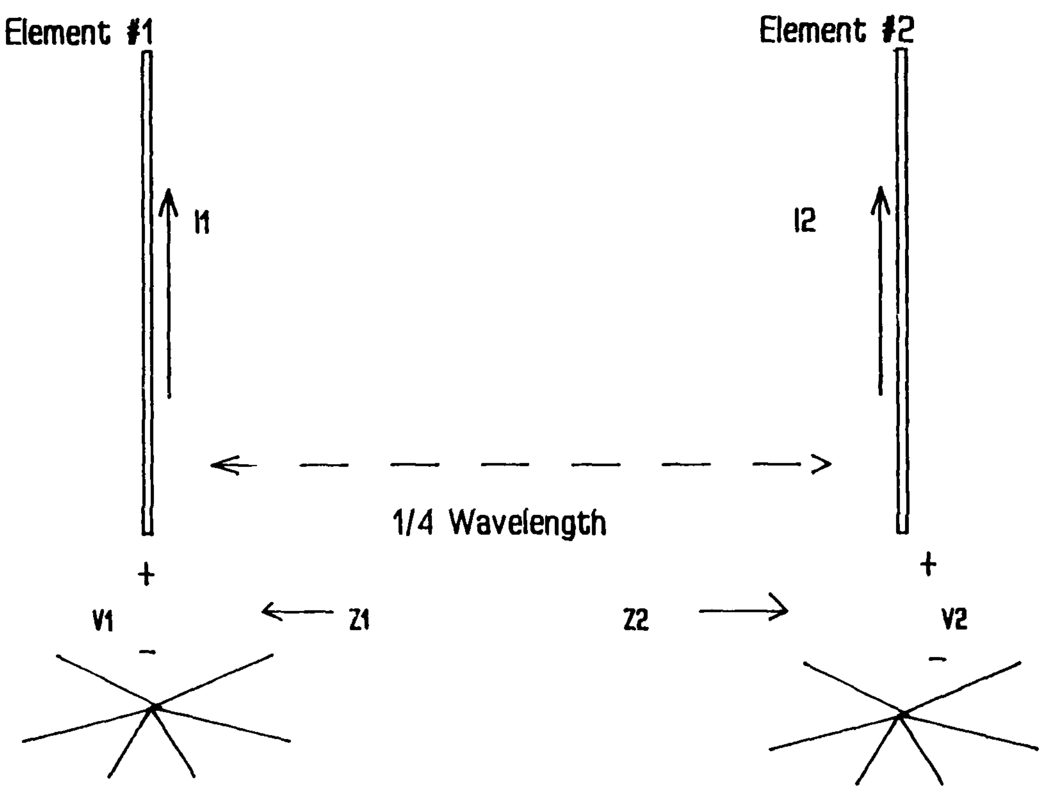


FIGURE 20 - Two Element Array

equations 3.13 and 3.14 in the following manner;

$$P_1 = |I_1|^2 R_{L1} = 1*\text{Real}[Z_{L1}] = 1*50.2 = 50.2 \text{ watts} \qquad (3.21)$$

$$P_2 = |I_2|^2 R_{L2} = 1*\text{Real}[Z_{L2}] = 1*41.4 = 41.4 \text{ watts} \qquad (3.22)$$

Once the power to each network is known the input resistance to each antenna can be found using equation 3.18 as shown below.

$$R_1 = \frac{|Vin|^2}{P_1} \qquad R_2 = \frac{|Vin|^2}{P_2}$$

where

$$Vin = \sqrt{50*(P_1 + P_2)} = \sqrt{50*91.6} = 67.5 \qquad (3.23)$$

So

$$R_1 = 91.235 \text{ ohms} \qquad R_2 = 110.628 \text{ ohms}$$

The phase shift between the input and output port of the PI networks can be calculated by converting the complex values for $Z_{L1}$ and $Z_{L2}$ into phasor form. The driving point impedances are now

$$Z_{L1} = \sqrt{(51.2)^2 + (41.4)^2} \angle \mathrm{Tan}^{-1} \frac{41.4}{51.2}$$

$$Z_{L2} = \sqrt{(41.4)^2 + (51.2)^2} \angle \mathrm{Tan}^{-1} \frac{51.2}{41.4}$$

with

$$Z_{L1} = 65.8 \angle 39.5 \qquad Z_{L2} = 65.8 \angle 50.4$$

From the previous calculations the values for $R_1$, $R_2$, $Z_{L1}$, $Z_{L2}$ are now known. The load impedances are converted into their equivalent parallel circuits and the parallel resistance components are be plugged into equations (3.1 - 3.3) for $R_2$ in order to solve for the reactive components of the PI networks. The parallel reactive components are coupled with the parallel branch of the PI networks and the resulting reactance becomes the value of the particular parallel branch on the network. Substitution of $R_1$ and $Z_{L1}$ for $R_1$ and $R_2$ of equations 3.1, 3.2, and 3.3 results in

$$Z_{A1} = j\frac{(91.2)(124.01)\mathrm{Sin}(39.5)}{(124.01)(\mathrm{Cos}(39.5)) - (124.01)(91.2)}$$

$$Z_{B1} = j \frac{(91.2)(124.01)\text{Sin}(39.5)}{(91.2)(\text{Cos}(39.5)) - (124.01)(91.2)}$$

$$Z_{C1} = j\ (124.01)(91.2)\ \text{Sin}(39.5)$$

upon completion of the computations above

$$Z_{A1} = -j674.98$$

$$Z_{B1} = -j302.46$$

$$Z_{C1} = j67.65$$

The substitution of R2 and $Z_{L2}$ of the other PI network for R1 and R2 in equations 3.1, 3.2, and 3.3 take the following form

$$Z_{A2} = j \frac{(110.6)(69.56)\text{Sin}(50.4)}{(69.56)(\text{Cos}(50.4)) - (69.56)(110.6)}$$

$$Z_{B2} = j \frac{(110.6)(69.56)\text{Sin}(50.4)}{(110.6)(\text{Cos}(50.4)) - (69.56)(110.6)}$$

$$Z_{C2} = j\ (69.56)(110.6)\text{Sin}(50.4)$$

The above computations result in

$$Z_{A2} = -j136.67$$

$$Z_{B2} = -j259.56$$
$$Z_{C2} = j67.58$$

The inductance and capacitance values corresponding to calculated reactance values for the paramaters of the two PI networks can be found assuming a frequency of 7 MHz with the following equations:

$$C = \frac{1}{wX_C} \qquad L = \frac{X_L}{w}$$

where

$$X_L = \text{a positive } j(\text{ ZA or ZB or ZC})$$
$$X_C = \text{a negative } j(\text{ ZA or ZB or ZC})$$
$$w = 2\pi f \ (\ f = \text{frequency})$$

Therefore, the PI network parameters for the two networks are

$$C_{ZA1} = 33.68\text{pF} \qquad C_{ZA2} = 166.7\text{pF}$$
$$C_{ZB1} = 75.64\text{pF} \qquad C_{ZB2} = 87.6\text{pF}$$
$$L_{ZC1} = 1.53\text{uH} \qquad L_{ZC2} = 1.53\text{uH}$$

and the networks are shown in Figure 21.

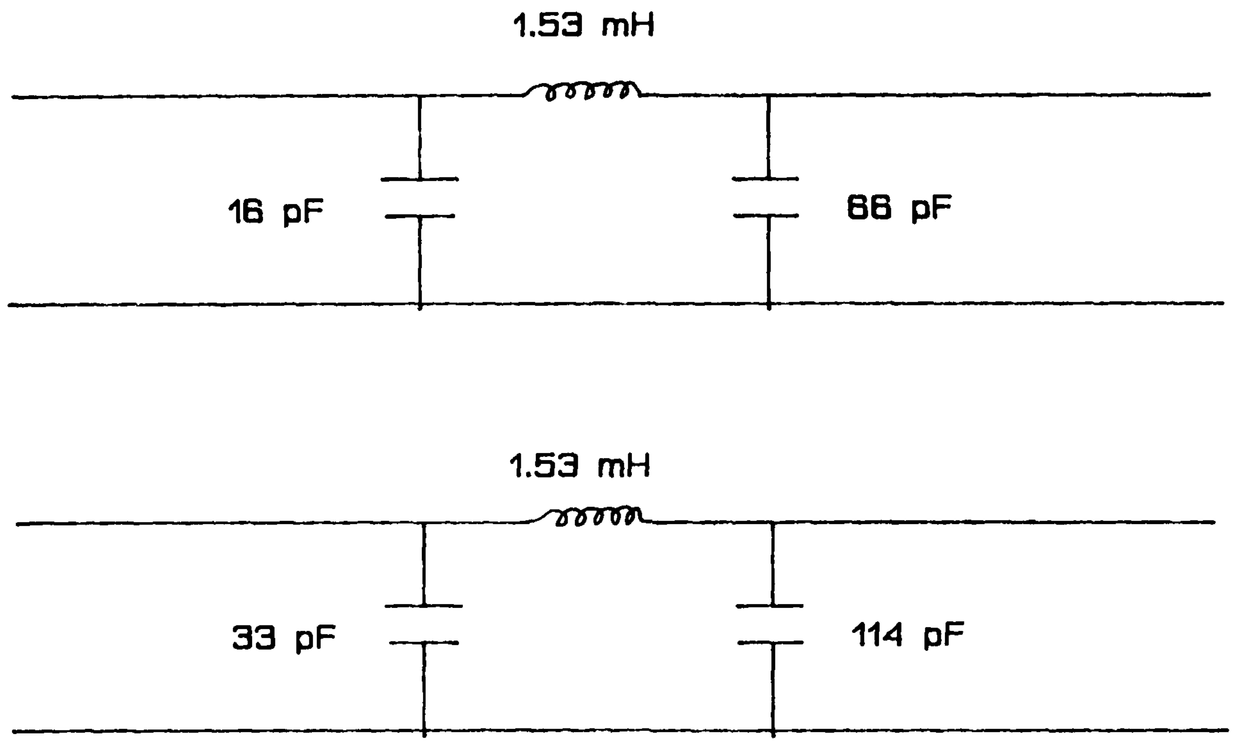


FIGURE 21 - PI Networks of 2 Element Antenna System

## D. Conclusion

This chapter listed many of the currently acceptable methods used for impedance matching in phased array antenna systems. Discussion of PI and TEE matching networks was presented in some detail for two reasons: PI and TEE networks are an often used form of impedance matching, and the PI network will be used in the equilateral array studied in this thesis as the circuit for controlling the radiation pattern of the array. A simple two element array with series PI networks was used to correctly match the antennas feed point impedances with the impedances being seen from the transmitter side of the network looking towards the individual element.

# IV. ANALYSIS OF A THREE ELEMENT PHASED ARRAY

A three element equilateral triangular phased array antenna system was chosen to test the two design methods developed in Chapters Five and Seven because of the comparatively small number of radiating elements involved. In this chapter the analysis of the three element triangular array will be presented.

The "analysis" procedure to be demonstrated in this chapter, as well as the "design" procedures of Chapters Five and Seven, can be extended to larger arrays. Figure 22 shows an NxN element array typical of larger designs.

The three elements of the equilateral array used throughout this thesis are quarter-wavelength vertical antennas spaced a quarter-wavelength apart. The coaxial transmission line connected between the pairs of elements and in series with the PI networks has a characteristic impedance of 50 ohms. The array is excited by a 50-ohm transmission line from the transmitter to element #1. Parameters of the PI networks are assumed to be purely reactive in all calculations.

## A. Transmission Line Equations

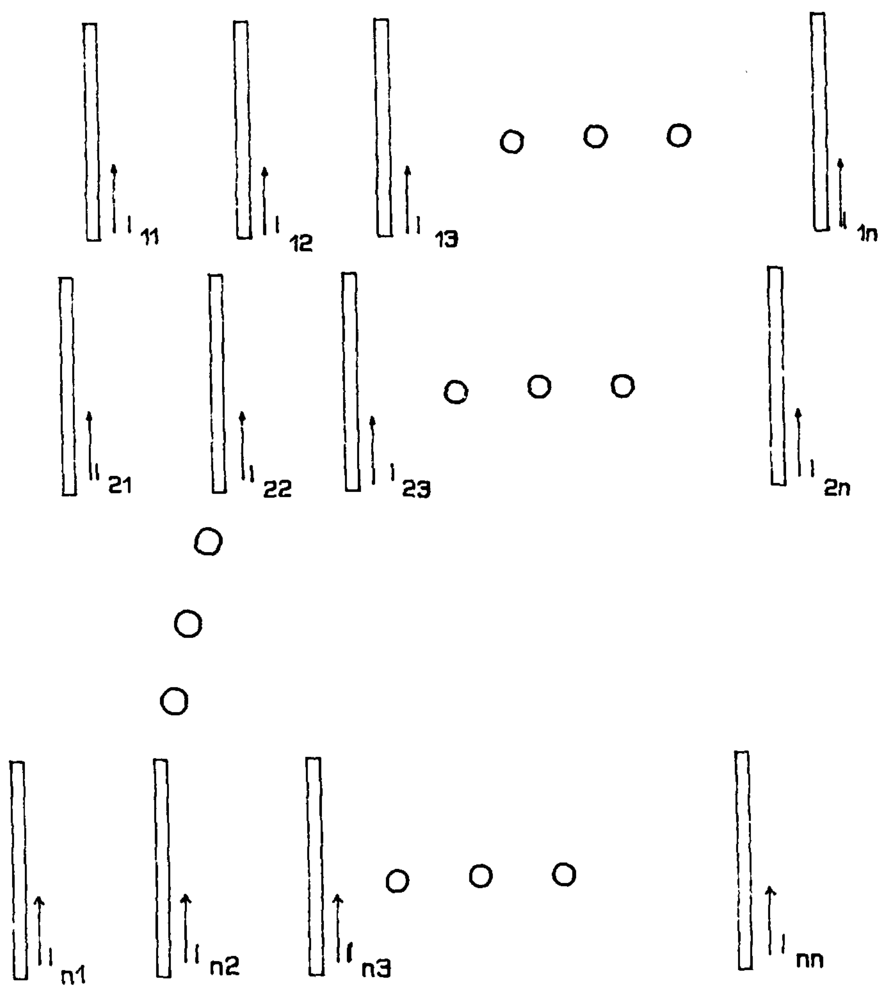


FIGURE 22 - NxN Element Array

Coaxial transmission lines are inherently lossy. The physical characteristics of the lossy transmission lines can be described as follows:[24]

$$V_s = V_r \text{Cosh}\sqrt{ZYL} + I_r Z_o \text{Sinh}\sqrt{ZYL} \qquad (4.1)$$

$$I_s = I_r \text{Cosh}\sqrt{ZYL} + \frac{V_r}{Z_o} \text{Sinh}\sqrt{ZYL} \qquad (4.2)$$

$Z = R + jwL$ (series impedance per unit length)
$Y = G + jwC$ (shunt admittance per unit length)

$V_r$ = receiving-end voltage (load)
$V_s$ = sending-end voltage (generator)

$I_r$ = receiving-end current (load)
$I_s$ = sending-end current (generator)

| | |
|---|---|
| R = reactance | L = inductance |
| C = cpacitance | G = leakage |

However, for some communications applications in the amateur amd broadcast radio fields, the transmission lines can be assumed to be lossless because the actual losses in the lines are quite small. For lossless transmission lines, equations (4.1 and 4.2) take on the following form:[25]

$$Z_o = \text{Characteristic Impedance} = 50+j0 \text{ ohms} \qquad (4.3)$$

$$V_s = V_r \text{Cos}\frac{2\pi L}{\lambda} + jI_r Z_o \text{Sin}\frac{2\pi L}{\lambda} \qquad (4.4)$$

$$I_s = I_r \mathrm{Cos}\, \frac{2\pi L}{\lambda} + j\frac{V_r}{Z_o}\mathrm{Sin}\, \frac{2\pi L}{\lambda} \qquad (4.5)$$

Equations of the above form will describe each segment of transmission line used in the interconnection networks.

## B. Mutual Impedance Equations

An antenna with current flowing in it will induce a voltage in any other antennas in the nearby area. The antennas will behave as though they were coupled to each other and the induced voltage in a second antenna divided by the current flowing in the excited antenna represents a mutual impedance. Denoting the mutual impedance between elements 1 and 2 by $Z_{12}$, an alternative form of equation 3.7 can be written as

$$Z_{12} = \frac{V_1}{I_2} - \frac{I_1 Z_{11}}{I_2} \qquad (4.6)$$

where $E_2$ is the induced voltage in the second antenna and $I_1$ is the excitation current in the first antenna (see Figure 21). The mutual impedance is expressed in terms of the current at the base of the first antenna and the voltage (induced by the current in the first antenna) at the base of the second antenna. This mutual impedance between two elements also varies with the geometry of the antenna and geometry of the array. The voltage and current relationships in arrays of elements can be represented by summing the voltage generated in the nth antenna caused by currents in all n antennas (extension of equations 3.7 and 3.8), as follows:

$$V_1 = I_1Z_{11} + I_2Z_{12} \text{-----} I_nZ_{1n} \tag{4.7}$$

$$V_2 = I_1Z_{12} + I_2Z_{22} \text{-----} I_nZ_{2n} \tag{4.8}$$

$$\begin{matrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{matrix}$$

$$V_n = I_nZ_{1n} + I_2Z_{2n} \text{-----} I_nZ_{nn} \tag{4.9}$$

where

$V_n$ = voltage applied to the base of element n.

$I_n$ = base current flowing in antenna n.

$Z_{nn}$ = self impedance of element n.

$Z_{ij}$ = mutual impedance between antennas i and j.

The mutual impedance ($Z_{12}$) between two antennas is defined as the ratio of the voltage at the base of the second antenna to the current flowing at the base of the first antenna. The self and mutual impedance values representing the elements of the equilateral triangular array are calculated from theory, using an assumed current excitation on elements of the same geometry and spacing as the array.[26] Mutual impedance measurements are difficult, as they are easily affected by the surroundings in the area of the array and the conductivity of the earth's surface. Therefore, these values should not be assumed accurate and measurements should be made at the actual constructed array with the measured values being used in the re-evaluation of any calculations.

## C. PI Network Equations

The PI network shown in Figure 4 is the form to be used in the interconnection medium of the phased array antenna system of Figure 5. The PI network is connected in series with two short sections of transmission line between each pair of elements in the array. Kirchhoff's Voltage Law is used in the analysis of the PI network where

$$\frac{1}{sC} = \frac{1}{jwC} = \frac{-j}{wC}, \qquad \text{with } X_C = \frac{1}{wC}$$

$$sL = jwL = jX_L, \qquad \text{with } X_L = wL$$

L = inductance C = capacitance

$X_L$ = reactive inductane in ohms
$X_C$ = reactive capacitance in ohms

The KCL equations at nodes #1 and #2 then can be written as

$$I_{13A} = \frac{V_{1A}}{(1/jX_{C13})} + \frac{(V_{1A} - V_{3A})}{jX_{L13}} \qquad (4.10)$$

$$I_{31A} = \frac{V_{3A}}{(1/jX_{C31})} + \frac{(V_{3A} - V_{1A})}{jX_{L13}} \qquad (4.11)$$

After transferring $X_{C13}$ and $X_{C31}$ to the numerator of the second component, equations (4.10) and (4.11) become

$$I_{13A} = V_{1A}(jX_{C13}) + \frac{(V_{1A} - V_{3A})}{jX_{L13}} \qquad (4.12)$$

$$I_{31A} = V_{3A}(jX_{C31}) + \frac{(V_{3A} - V_{1A})}{jX_{L13}} \qquad (4.13)$$

The currents ($I_{13A}$ and $I_{31A}$) and voltages ($V_{1A}$ and $V_{3A}$) in (4.12) and (4.13) are complex quantities, but the impedance representations for the inductor and capacitors are only shown with the imaginary components ($-jX_{L13}$, $jX_{C13}$, $jX_{C31}$) because the real components are assumed to be zero; that is, only lossless networks are being considered here.

Combining the transmission line equations, the effects of mutual impedance between the elements, the Kirchhoff's Current Law equation at the base of each antenna, and the PI network equations yields a set of 24 complex equations describing the entire array as follows:

Mutual Impedance equations:

$$V_1 = I_1Z_{11} + I_2Z_{12} + I_3Z_{13} \tag{4.14}$$

$$V_2 = I_1Z_{21} + I_2Z_{22} + I_3Z_{23} \tag{4.15}$$

$$V_3 = I_1Z_{31} + I_2Z_{32} + I_3Z_{33} \tag{4.16}$$

Transmission Line equations:

$p=(2\pi L)*3/2, \quad L=1/8, \quad Zo=50.0$

$$V_1 = V_{1A}\mathrm{Cos}(p) + jZ_oI_{13A}\mathrm{Sin}(p) \tag{4.17}$$

$$I_{13} = I_{13A}\mathrm{Cos}(p) + \frac{j}{Z_o} V_{1A}\mathrm{Sin}(p) \tag{4.18}$$

$$V_3 = V_{3A}\mathrm{Cos}(p) + jZ_oI_{31A}\mathrm{Sin}(p) \tag{4.19}$$

$$I_{31} = I_{31A}\mathrm{Cos}(p) + \frac{j}{Z_o} V_{3A}\mathrm{Sin}(p) \tag{4.20}$$

$$V_1 = V_{1B}\mathrm{Cos}(p) + jZ_oI_{12B}\mathrm{Sin}(p) \tag{4.21}$$

$$I_{12} = I_{12B}\mathrm{Cos}(p) + \frac{j}{Z_o} V_{1B}\mathrm{Sin}(p) \tag{4.22}$$

$$V_2 = V_{2B}\mathrm{Cos}(p) + jZ_oI_{21B}\mathrm{Sin}(p) \qquad (4.23)$$

$$I_{21} = I_{21B}\mathrm{Cos}(p) + \frac{j}{Z_o} V_{2B}\mathrm{Sin}(p) \qquad (4.24)$$

$$V_2 = V_{2C}\mathrm{Cos}(p) + jZ_oI_{23C}\mathrm{Sin}(p) \qquad (4.25)$$

$$I_{23} = I_{23C}\mathrm{Cos}(p) + \frac{j}{Z_o} V_{2C}\mathrm{Sin}(p) \qquad (4.26)$$

$$V_3 = V_{3C}\mathrm{Cos}(p) + jZ_oI_{32C}\mathrm{Sin}(p) \qquad (4.27)$$

$$I_{32} = I_{32C}\mathrm{Cos}(p) + \frac{j}{Z_o} V_{3C}\mathrm{Sin}(p) \qquad (4.28)$$

Kirchhoff's Current Law (at base of each antenna):

$$I_1 + I_{12} + I_{13} = I_T \qquad (4.29)$$

$$I_2 + I_{21} + I_{23} = 0 \qquad (4.30)$$

$$I_3 + I_{31} + I_{32} = 0 \qquad (4.31)$$

PI Network equations:

$$I_{13A} = V_{1A}(jX_{C13}) + \frac{(V_{1A}-V_{3A})}{jX_{L13}} \qquad (4.32)$$

$$I_{31A} = V_{3A}(jX_{C31}) + \frac{(V_{3A}-V_{1A})}{jX_{L13}} \qquad (4.33)$$

$$I_{12B} = V_{1B}(jX_{C12}) + \frac{(V_{1B}-V_{2B})}{jX_{L12}} \qquad (4.34)$$

$$I_{21B} = V_{2B}(jX_{C21}) + \frac{(V_{2B}-V_{1B})}{jX_{L12}} \qquad (4.35)$$

$$I_{23C} = V_{2C}(jX_{C23}) + \frac{(V_{2C}-V_{3C})}{jX_{L23}} \qquad (4.36)$$

$$I_{32C} = V_{3C}(jX_{C32}) + \frac{(V_{3C}-V_{2C})}{jX_{L23}} \qquad (4.37)$$

In order to simplify the implementation of these equations into FORTRAN code, the transcendental components of the transmission line equations are (from this point on) represented in the following form:

$$K_1 = \text{Cos}\,\frac{2\pi L}{\lambda} \qquad K_2 = Z_o\text{Sin}\,\frac{2\pi L}{\lambda} \qquad K_3 = \text{Sin}\,\frac{2\pi L}{Z_o\lambda}$$

The variables in this se+ of 24 complex equations are labeled on the equilateral phased array diagram in Figure 5. The variables $V_1$, $V_2$, $V_3$ are the voltage at the base of each of the three elements of the array. $I_1$, $I_2$, $I_3$ are defined as the current distributions (current flowing through each element). The variables $V_{1A}$, $V_{3A}$, $V_{1B}$, $V_{2B}$, $V_{2C}$, $V_{3C}$ are the voltages at the ends of the transmission line sections that are not connected to the base of the antenna, but to the PI networks. The $I_{12}$, $I_{21}$, $I_{13}$, $I_{31}$, $I_{23}$, $I_{32}$ variables are the currents flowing from the array elements through each section of transmission line towards the PI networks. The currents flowing at the end of the transmission line sections connected to the PI networks are the variables $I_{13A}$, $I_{31A}$, $I_{12B}$, $I_{21B}$, $I_{23C}$, $I_{32C}$. The last variable of this set of equations is $I_T$, which is defined as the input current from the source of power for the array and is input at element #1 of the array.

In the "analysis" of the equilateral array the reactive components of the PI networks are given as known values in the set of equations describing the system. The currents flowing in the elements of the array become the unknown variables to be calculated. Upon close examination of this linear set of equations the possibility of substituting one of the groups of equations into one of the other groups of equations is realized. Substitution of the mutual impedance and PI network equations into the transmission line equations was performed to bring about a reduction in the number of equations as well as in the complexity of the set of equations needing to be solved. Upon completion of the above mentioned simplification measures for the "analysis" situation, the 24 complex equations (4.14 thru 4.37) take the following complex <u>linear</u> form:

$$I_1 + I_{12} + I_{13} = I_T \qquad (4.38)$$

$$I_2 + I_{21} + I_{23} = 0 \qquad (4.39)$$

$$I_3 + I_{31} + I_{32} = 0 \qquad (4.40)$$

$$I_1Z_{11}+I_2Z_{12}+I_3Z_{13}-V_{1A}\left[K_1-K_2X_{C13}+\frac{K_2}{X_{L13}}\right]+ V_{3A}\frac{-K_2}{X_{L13}} = 0 \qquad (4.41)$$

$$I_{13}-V_{1A}\left[K_1jX_{C13} - \frac{jK_1}{X_{L13}} + jK_3\right]- V_{3A}\frac{jK_1}{X_{L13}} = 0 \qquad (4.42)$$

$$I_1Z_{31}+I_2Z_{32}+I_3Z_{33}-V_{1A}\frac{-K_2}{X_{L13}} + V_{3A}\left[K_1+ \frac{K_2}{X_{L13}} -K_2X_{C31}\right]=0 \qquad (4.43)$$

$$I_{31} - V_{1A}\frac{jK_1}{X_{L13}} - V_{3A}\left[K_1jX_{C32} - \frac{jK_1}{X_{L13}} +jK_3\right]= 0 \qquad (4.44)$$

$$I_1Z_{11}+I_2Z_{12}+I_3Z_{13}-V_{1B}\left[K_1-K_2X_{C12}+\frac{K_2}{X_{L12}}\right]+ V_{2B}\frac{-K_2}{X_{L12}} = 0 \qquad (4.45)$$

$$I_{13}-V_{1B}\left[K_1jX_{C12} - \frac{jK_1}{X_{L12}} + jK_3\right]- V_{2B}\frac{jK_1}{X_{L12}} = 0 \qquad (4.46)$$

$$I_1Z_{21}+I_2Z_{22}+I_3Z_{22}-V_{1B}\frac{-K_2}{X_{L12}} + V_{2B}\left[K_1+ \frac{K_2}{X_{L12}} -K_2X_{C21}\right]=0 \qquad (4.47)$$

$$I_{31} - V_{1B}\frac{jK_1}{X_{L13}} - V_{2B}\left[K_1jX_{C21} - \frac{jK_1}{X_{L12}} + jK_3\right] = 0 \qquad (4.48)$$

$$I_1Z_{21}+I_2Z_{22}+I_3Z_{23}-V_{2C}\left[K_1-K_2X_{C23}+\frac{K_2}{X_{L23}}\right]+V_{3C}\frac{-K_2}{X_{L23}} = 0 \qquad (4.49)$$

$$I_{13}-V_{2C}\left[K_1jX_{C23} - \frac{jK_1}{X_{L23}} + jK_3\right] - V_{3C}\frac{jK_1}{X_{L23}} = 0 \qquad (4.50)$$

$$I_1Z_{31}+I_2Z_{32}+I_3Z_{33}-V_{2C}\frac{-K_2}{X_{L23}} + V_{3C}\left[K_1+\frac{K_2}{X_{L23}} - K_2X_{C32}\right] = 0 \qquad (4.51)$$

$$I_{31} - V_{2C}\frac{jK_1}{X_{L23}} - V_{3C}\left[K_1jX_{C32} - \frac{jK_1}{X_{L23}} + jK_3\right] = 0 \qquad (4.52)$$

Therefore in the analysis case the unknown variables are

$I_{12}$, $I_{21}$, $I_{13}$, $I_{31}$, $I_{23}$, $I_{32}$, $V_{1A}$, $V_{3A}$, $V_{1B}$, $V_{2B}$, $V_{2C}$, $V_{3C}$, $I_1$, $I_2$, $I_3$

and the known variables are

$X_{C12}$, $X_{C21}$, $X_{C13}$, $X_{C31}$, $X_{C23}$, $X_{C32}$, $I_T$
$X_{L12}$, $X_{L13}$, $X_{L23}$

The above set of linear (since the network parameters are given) equations can be solved using matrix techniques for solving a set of linear simultaneous equations of the form Ax=b. Here A is a 15x15 complex matrix consisting of the coefficients of the unknown variables, and the complex matrix b consists of the only constant in the system (the array input current $I_T$). The PI network parameters are known but they are coeffiicents of the variable voltages at the ends of the transmission line sections connected to the PI networks ($V_{1A}$, ..., $V_{3C}$) in the 15 linear equations (4.38 thru 3.52).

A FORTRAN subroutine was written which solves equations 4.38 thru 4.52 using a linear equation solver routine (LEQ2C) located in the IMSL subroutine library available on the VAX 11/780. This analysis subroutine ANAL_DESG returns to the user the complex current distributions of the array elements after being supplied the array's input current ($I_T$) and the complex PI network parameters. The source code for this subroutine CURR and the main program ANAL_DESG are listed Appendix B.

## D. Results

An example run of this subroutine is shown below; it was supplied with constant magnitudes for all the PI network parameters and a value for the input current $I_T$.

$X_{L13}$ = 1 $\quad$ $X_{C13}$ = -50 $\quad$ $X_{C31}$ = -560

$X_{L12}$ = 10 $\quad$ $X_{C12}$ = -0.01 $\quad$ $X_{C21}$ = -50

$X_{L23}$ = 100 $X_{C23}$ = -25 $X_{C32}$ = -5

$I_T$ (input current) = 1.0 + j0

Resulting Magnitude and Phase

| | | |
|---|---|---|
| $M_{I1}$ = 1.0 | $M_{I2}$ = 0.28 | $M_{I3}$ = 0.28 |
| $PH_{I1}$ = 0.0 | $PH_{I2}$ = 95.6 | $PH_{I3}$ = 95.58 |

The values for the PI network paramters in the above example were choosen arbitrarily and thus do not necessarily result in current distributions that correspond to an useful radiation pattern. The radiation pattern corresponding to the current distributions found above (I1,I2, & I3) is shown in Figure 23.

Below is a list of two more examples showing the given PI network parameters and the resulting current distributions.

| | | |
|---|---|---|
| $X_{L13}$ = 100 | $X_{C13}$ = -50 | $X_{C31}$ = -500 |
| $X_{L12}$ = 100 | $X_{C12}$ = -5.67 | $X_{C21}$ = -50 |
| $X_{L23}$ = 100 | $X_{C23}$ = -40 | $X_{C32}$ = -500 |

IT (input current) = 1.0 + j0

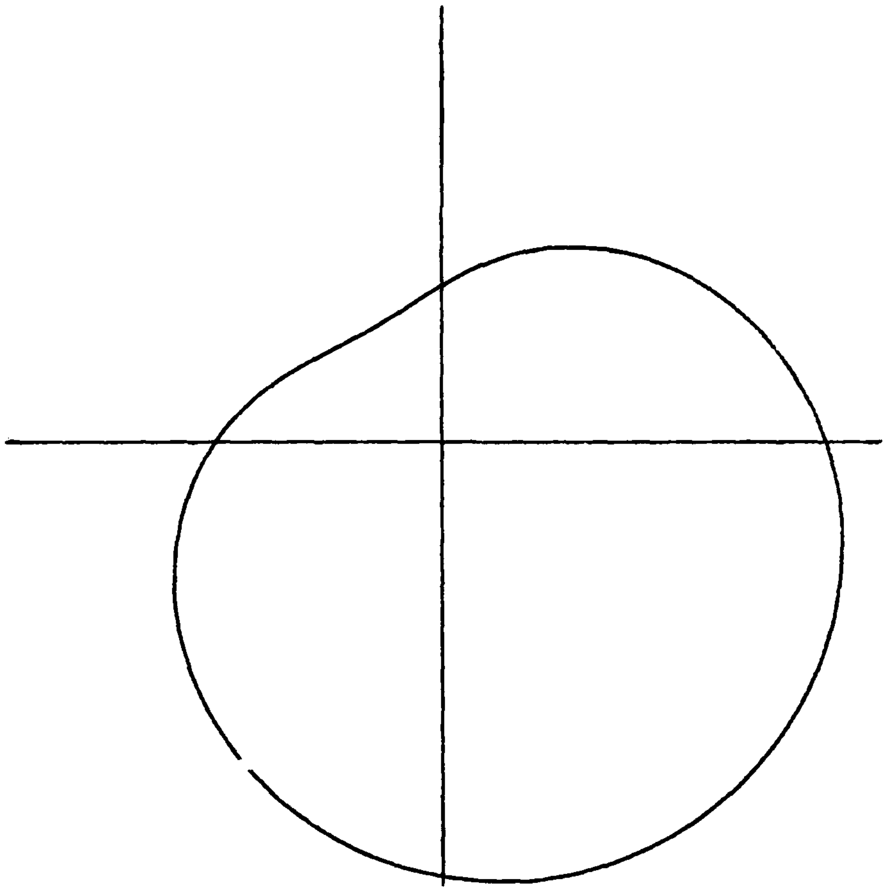


FIGURE 23 - Pattern from Analysis Results

Resulting Magnitude and Phase

| | | |
|---|---|---|
| $M_{I1} = 1.0$ | $M_{I2} = 0.28$ | $M_{I3} = 0.28$ |
| $PH_{I1} = 0.0$ | $PH_{I2} = 95.48$ | $PH_{I3} = 95.48$ |

| | | |
|---|---|---|
| $X_{L13} = 10$ | $X_{C13} = -5$ | $X_{C31} = -50$ |
| $X_{L12} = 15$ | $X_{C12} = -0.01$ | $X_{C21} = -10$ |
| $X_{L23} = 20$ | $X_{C23} = -0.02$ | $X_{C32} = -5$ |

IT (input current) = 100.0 + j45

Resulting Magnitude and Phase

| | | |
|---|---|---|
| $M_{I1} = 1.0$ | $M_{I2} = 0.11$ | $M_{I3} = 0.28$ |
| $PH_{I1} = 0.0$ | $PH_{I2} = 80.67$ | $PH_{I3} = 85.72$ |

## E. Summary

This chapter has demonstrated that the phased array antenna system represented in Figure 5 can be analyzed by solving the 15 linear simultaneous equations that describe the system. The current distribution for each element of the array are the three variables of most concern resulting from the solution to equations 4.38 to 4.52. However, these resulting current distributions are as arbitrary (useless) as the corresponding PI network parameters specified by the user.

Therefore, in order to design a phased array antenna system based on the interconnection medium illustrated in Figure 5, the 24 complex equations that describe the physical and electrical characteristic of the system must be solved using specified (desired) current distribution. These current distributions will correspond to a desired radiation pattern. This idea is explored in Chapter Five.

Presentation and analysis of the equations describing the physical and electrical characteristics of the three element equilateral triangular array was the main theme of Chapter Four. The substitutions of the mutual impedance and PI network equations into the transmission line equations was accomplished and the resulting set of 15 equations was presented in their complex linear form, with the known and unknown variables clearly indicated. All of the necessary equations and theories has been discussed and the stage is now set for developement and discussion of the PI network design procedures to be presented in Chapters Five and Seven.

# V. INTERCONNECTION NETWORK DESIGN BY LINEARIZATION: AN EXTENSION OF NEWTON'S METHOD

This chapter will present a detailed explanation of a procedure capable of solving the equations that describe the unique phased array antenna system of Figure 5 in the "design" situation. As mentioned in the introduction, the design process pertains to the situation where the currents on the radiating elements are classified as given (known constant complex quantities), while the PI network parameters are the unknown variables of interest. Therefore, the "design" of interconnection networks for the array is just the opposite, where the variables of interest are concerned, of the "analysis" of interconnection networks that was presented at the end of Chapter Four.

To a potential user of the equilateral array proposed in this thesis, the "design" situation would be of most interest. The user probably already has calculated the current distributions on the elements of the particular array (discussed in Chapter Two) and would need only the proper PI network parameters to realize the desired radiation pattern. The design procedure of this chapter provides these PI network parameters given the current distributions on the elements as input.

The discussion will begin with an explanation of the substitutions made among the 24 equations describing the array (equations 4.14 thru 4.37), justification for giving some variables a constant value, and why one of the Kirchhoff's Current Law equations was disregarded (not used) as an equation describing the array in this design procedure. The nonlinear equations are presented in their complex form and a detailed discussion of the linearization process performed on these equations follows. The linearized set of complex design equations and the results of an implementation of this design method on the equilateral triangular array, with $X_{C12}$ and $X_{C23}$ being the unknown variables, is shown as an example.

In this design method the parameters of the PI networks are unknown values. The desired current for each element in the array will be input as known values from the program MAX_RAD (discussed in Chapter Two and listed in Appendix A). Reducing the amount of equations and the number of unknown variables by substitution of the mutual impedance and PI network equations into the transmission line equations was done in order to simplify the computational procedure.

Substituting the mutual impedance equations into the tansmission line equations for the variables $V_1$, $V_2$, and $V_3$ and the PI network equations for the variables $I_{13A}$, $I_{31A}$, $I_{12B}$, $I_{21B}$, $I_{23C}$, and $I_{32C}$ yields a set of 12 equations. Together with Kirchhoff's Current Law (KCL) equations, the number of equations now describing the system has been dropped to 15. The voltage at the base of each antenna ($V_1$, $V_2$, $V_3$) are constants for this method since the input currents $I_1$, $I_2$, and $I_3$ are given and the self and mutual impedance values $Z_{11}$, $Z_{12}$, and $Z_{13}$ are also known values. Thus, equations (4.7-4.9) give $V_1$, $V_2$, and $V_3$ directly. In order to solve these 15 equations (4.38 thru 4.52) simultaneously there must be the same number of equations as there are unknowns. The current flowing into the array from the transmitter ($I_T$) is an unknown variable that must be disregarded in this design procedure. So equation 4.38, which is the KCL equation at the base of antenna #1, is taken out of the system in order to balance the number of equations with the number of unknown variables. Thus, equation 4.38 is assumed non-existent and is not included in the set of 14 nonlinear design equations describing the array. There remains 14 equations with 21 unknown variables.

In order to make the 14 equations solvable, seven of the nine PI network parameters were assigned specified values and so they become knowns, yielding 14 nonlinear complex equations in 14 unknowns. The PI network parameters that were not given a value become the unknown variables of interest. For the example shown in this chapter $X_{C12}$ and $X_{C23}$ are the unknown network paramaters.

## A. Design Equations

The 14 nonlinear complex equations have been arranged in a form that is compatible with the linear equation solver routine (LEQ2C) of the IMSL subroutine library. The design equations are listed below in this form:

$$I_{21} + I_{23} = -I_2 \tag{5.1}$$

$$I_{31} + I_{32} = -I_3 \tag{5.2}$$

$$V_{1A}\left[K_1 - K_2X_{C13} + \frac{K_2}{X_{L13}}\right] + V_{3A}\frac{-K_2}{X_{L13}} = V_1 \tag{5.3}$$

$$I_{13} - V_{1A}\left[K_1jX_{C13} - \frac{jK_1}{X_{L13}} + jK_3\right] - V_{3A}\frac{jK_1}{X_{L13}} = 0 \tag{5.4}$$

$$V_{1A}\frac{-K_2}{X_{L13}} + V_{3A}\left[K_1 + \frac{K_2}{X_{L13}} - K_2X_{C31}\right] = V_3 \tag{5.5}$$

$$I_{31} - V_{1A}\frac{jK_1}{X_{L13}} - V_{3A}\left[K_1jX_{C31} - \frac{jK_1}{X_{L13}} + jK_3\right] = 0 \tag{5.6}$$

$$V_{1B}\left[K_1 - K_2X_{C12} + \frac{K_2}{X_{L12}}\right] + V_{2B}\frac{-K_2}{X_{L12}} = V_1 \tag{5.7}$$

$$I_{12} - V_{1B}\left[K_1jX_{C12} - \frac{jK_1}{X_{L12}} + jK_3\right] - V_{2B}\frac{jK_1}{X_{L12}} = 0 \tag{5.8}$$

$$V_{1B}\frac{-K_2}{X_{L12}} + V_{2B}\left[K_1 + \frac{K_2}{X_{L12}} - K_2X_{C21}\right] = V_2 \qquad (5.9)$$

$$I_{21}-V_{1B}\frac{jK_1}{X_{L12}} - V_{2B}\left[K_1jX_{C21} - \frac{jK_1}{X_{L12}} + jK_3\right] = 0 \qquad (5.10)$$

$$V_{2C}\left[K_1 - K_2X_{C23} + \frac{K_2}{X_{L23}}\right] + V_{3C}\frac{-K_2}{X_{L23}} = V_2 \qquad (5.11)$$

$$I_{23}-V_{2C}\left[K_1jX_{C23} - \frac{jK_1}{X_{L23}} + jK_3\right] - V_{3C}\frac{jK_1}{X_{L23}} = 0 \qquad (5.12)$$

$$V_{2C}\frac{-K_2}{X_{L23}} + V_{3C}\left[K_1 + \frac{K_2}{X_{L23}} - K_2X_{C32}\right] = V_3 \qquad (5.13)$$

$$I_{32}-V_{2C}\frac{jK_1}{X_{L23}} - V_{3C}\left[K_1jX_{C32} - \frac{jK_1}{X_{L23}} + jK_3\right] = 0 \qquad (5.14)$$

Since the unknown PI network parameter are $X_{C12}$ and $X_{C23}$ the unknown variables of the design equations listed above are:

$I_{12}$, $I_{21}$, $I_{23}$, $I_{32}$, $I_{13}$, $I_{31}$, $V_{1A}$, $V_{3A}$, $V_{1B}$, $V_{2B}$, $V_{2C}$, $V_{3C}$, $X_{C12}$, $X_{C23}$

and the known variables are

$V_1$, $V_2$, $V_3$, $I_1$, $I_2$, $I_3$, $Z_{11}$, $Z_{12}$, $Z_{13}$, $Z_{21}$, $Z_{22}$,

$Z_{23}$, $Z_{31}$, $Z_{32}$, $Z_{33}$, $X_{L12}$, $X_{C21}$, $X_{L13}$, $X_{C13}$, $X_{C31}$
$X_{L23}$, $X_{C32}$

The $K_1$, $K_2$, and $K_3$ terms are defined as they were in Chapter Four on page 69.

Upon observation of the 14 design equations one can see the nonlinearity is due to the occurrence of crossproducts between the unknown phasor voltages and the unknown PI network parameters. For instance, since $X_{C12}$ is an unknown network parameter that needs to be found, an example of a nonlinear component is the $V_{1B}X_{C12}$ term in equations 5.7 and 5.8. Also the component $V_{1A}X_{C13}$ is nonlinear in equations 5.4 and 5.5 if the $X_{C13}$ network parameter was an unknown capacitor of interest.

## B. Taylor's Series Expansion

In order to solve the 14 design equations exactly, one must use a linear equation solver routine which utilizes matrix algebra techniques (Ax=b). Therefore the nonlinear equations must be linearized. The method used to do this is a generalization of Newton's method for finding the roots of f(x)=0.[27] The generalization involves extension to the multivariable, complex variable case presented by the solution of equations 5.4 thru 5.15.

The extension is as follows.

Transposing all terms of equations 5.4 thru 5.15 to the left of the equal sign, the design relations may be written as

$$f_i(x_1,\ldots,x_n) = 0 \quad i=1, \ldots, n \qquad (5.16)$$

where $f_i(x_1,\ldots,x_n)$ is the complex left hand side of the ith equation and 0 denotes the complex number 0 + j0. $\underline{x} = (x_1,\ldots,x_n)$ *denotes the complex* (Real + jReal) problem unknowns, whether voltages, currents, or parameter values. Expanding $f_i$ in a complex n-dimensional Taylor's series about an initial (complex) guess $\underline{x}_o$ and retaining linear terms, one gets

$$f_i(x_1,\ldots,x_n)=f_i(x_{1o},\ldots,x_{no}) + \sum_{j=1}^{n} \left.\frac{\partial f_i}{\partial x_j}\right|_{\underline{x}_o}(x_j - x_{jo}) \qquad (5.17)$$

Equating the right side of equation 5.17 to 0+j0 (complex) yields

$$f_i(x_{1o},\ldots,x_{no}) + \sum_{j=1}^{n} \left.\frac{\partial f_i}{\partial x_j}\right|_{\underline{x}_o}(x_j - x_{jo}) = 0 \qquad (5.18)$$

or

$$f_i(x_{1o},\ldots,x_{no}) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}\bigg|_{\underline{x}_o} (x_j)$$

$$- \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}\bigg|_{\underline{x}_o} (x_{jo}) = 0 \qquad (5.19)$$

leading to a set of n simultaneous complex linear equations

$$\sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}\bigg|_{\underline{x}_o} (x_j) = \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}\bigg|_{\underline{x}_o} (x_{jo}) - f_i(x_{1o},\ldots,x_{no}) \quad i=1,\ldots n \qquad (5.20)$$

Equation (5.20) maybe expressed in complex vector/matrix form as

$$C(\underline{x}_o)\underline{x} = \underline{d}(\underline{x}_o) \qquad (5.21)$$

where $C(\underline{x}_o)$ is an nxn matrix of complex coefficents dependent upon $\underline{x}_o=(x_{1o},\ldots, x_{no})$ as the point of linearization, where

$$C_{ij}(\underline{x}_o) = \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}\bigg|_{\underline{x}_o} \quad i,j=1,\ldots,n \qquad (5.22)$$

and $\underline{d}(\underline{x}_o)$ is a complex n-vector whose elements are also dependent on $\underline{x}_o$:

$$d_i = \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}\bigg|_{\underline{x}_o} (x_{jo}) - f_i(x_{1o},\ldots,x_{no}) \quad i=1,\ldots,n \qquad (5.23)$$

From an initial complex vector guess $\underline{x}_o = (x_{1o},\ldots,x_{no})$ at the root of 5.16 may be solved via complex matrix inversion with

$$\underline{x} = (x_1,\ldots,x_n) = C^{-1}(\underline{x}_o)\underline{d}(\underline{x}_o) \qquad (5.24)$$

representing the approximate vector "root" of the original set of design equations 5.16.

Utilizing the approximate "root" as a new linearization point for the Taylor's series expansion, an iterative process determining a sequence of approximations to the solution of 5.16 is obtained:

$$\underline{x}_{k+1} = [C(\underline{x}_k)]^{-1}\,\underline{d}(\underline{x}_k) \qquad (5.25)$$

where the elements of $C(\underline{x})$ and $\underline{d}(\underline{x})$ are as given above. Essentially, the algorithm is a vector/matricized generalization of Newton's method to the multi(complex)variable case.

Any of the 14 design equations containing the two unknown PI network parameters must be linearized using the procedure discussed in the previous paragraphs.

The abstract explanation of the linearization given above is complemented by a listing of the linearization steps performed on the nonlinear equation (5.7).

$$F = V_{1B}(K_1 + K_2/X_{L12}) - V_{1B}X_{C12}K_2 - V_{2B}(K_2/X_{L12})$$

The Taylor Series Expansion begins with the step below.

$$F_o + \frac{\partial F}{\partial V_{1B}}(V_{1B}-V_{1Bo}) + \frac{\partial F}{\partial X_{C12}}(X_{C12}-X_{C12o}) + \frac{\partial F}{\partial V_{2B}}(V_{2B}-V_{2Bo})$$

or

$$F_o + A + B + C = V_1$$

after the partial derivatives are performed $F_o$, A, B, and C become

$$A = (K_1 + K_2/X_{C12} - K_2X_{C12o})(V_{1B} - V_{1Bo})$$
$$B = (-V_{1Bo}K_2)(X_{C12} - X_{C12o})$$
$$C = (-K_2/X_{L12})(V_{2B} - V_{2Bo})$$

$$F_o=V_{1Bo}(K_1 + K_2/X_{L12}) - V_{1Bo}X_{C12o}K_2 - V_{2Bo}(K_2/X_{L12})$$

Adding all the terms together (with a substantial number of them cancelling each other out) the remaining terms take the following form

$$V_{1B}\left[K_1 - K_2X_{C12} + \frac{K_2}{X_{L12}}\right] + V_{2B}\ \frac{-K_2}{X_{L12}}$$

$$+ X_{C12}(-V_{1Bo}K_2) = V_1 - X_{C12o}V_{1Bo}K_2$$

Upon completion of the linearization process on the 4 nonlinear equations, the 14 design equations take on the following form:

$$V_{1A}(K_1-K_2{*}X_{C13}+K_2/X_{L13}) + V_{3A}(-K_2/X_{L13}) = V_1 \quad (5.27)$$

$$I_{13}-V_{1A}(K_1jX_{C13}-jK_1/X_{L13}+jK_3)-V_{3A}(jK_1/X_{L13}) = 0 \quad (5.28)$$

$$V_{1A}(-K_2/X_{L13}) + V_{3A}(K_1+K_2/X_{L13}-K_2{*}X_{C31}) = V_3 \quad (5.29)$$

$$I_{31}-V_{1A}(jK_1/X_{L13})-V_{3A}(K_1jX_{C31}-jK_1/X_{L13}+jK_3) = 0 \quad (5.30)$$

$$V_{1B}\left[K_1 - K_2X_{C12} + \frac{K_2}{X_{L12}}\right] + V_{2B}\ \frac{-K_2}{X_{L12}}$$

$$+ X_{C12}(-V_{1Bo}K_2) = V_1 - X_{C12o}V_{1Bo}K_2 \quad (5.31)$$

$$I_{12} - V_{1B}\left[K_1jX_{C12o} - \frac{jK_2}{X_{L12}} +jK_3\right] - V_{2B}\ \frac{jK_1}{X_{L12}}$$

$$+ X_{C12}(-jK_1V_{1Bo}) = -jV_{1Bo}X_{C12o}K1 \quad (5.32)$$

$$V_{1B}(-K_2/X_{L12}) + V_{2B}(K_1+K_2/X_{L12}-K_2{*}X_{C21}) = V_2 \quad (5.33)$$

$$I_{21} - V_{1B}(jK_1/X_{L12})-V_{2B}(K_1jX_{C21}-jK_1/X_{L12}+jK_3) = 0 \quad (5.34)$$

$$V_{2C}\left[K_1-K_2{*}X_{C23}+\frac{K_2}{X_{L23}}\right]+V_{3C}\frac{-K_2}{X_{L23}}$$

$$+X_{C23}(-V_{2Co}K_2)=V_2-X_{C23o}V_{2Co}K_2 \qquad (5.35)$$

$$I_{23}-V_{2C}\left[K_1jX_{C23}-\frac{jK_1}{X_{L23}}+jK_3\right]-V_{3C}\frac{jK_1}{X_{L23}}$$

$$+X_{C23}(-jK_1V_{2Co})=-jV_{2Co}X_{C23o} \qquad (5.36)$$

$$V_{2C}(-K_2/X_{L23})+V_{3C}(K_1+K_2/X_{L23}-K_2{*}X_{C32})=V_3 \qquad (5.37)$$

$$I_{32}-V_{2C}(jK_1/X_{L23})-V_{3C}(K_1jX_{C32}-jK_1/X_{L23}+jK_3)=0 \qquad (5.38)$$

The coefficents of these 14 design equations are calculated using the known values, and these coefficents make up the matrix A in the formula $x = A^{-1}b$. Any resulting constants are put into the b matrix.

## C. Results

The design equations have been incorporated into a program named DESIGN which calculates the values for the $X_{C12}$ and $X_{C23}$ PI network design parameters. The program has been run with a range of input currents calculated using the beam steering method of Chapter Three and is listed in Appendix C. The input current distribution corresponding to a set of $\Theta_m$ and $\Theta_n$ (max and null) directions were calulated using the programs discussed in Chapter Two and stored in a data file. The design program is able to read in these currents and calulate the corresponding $X_{C12}$ and $X_{C23}$ values.

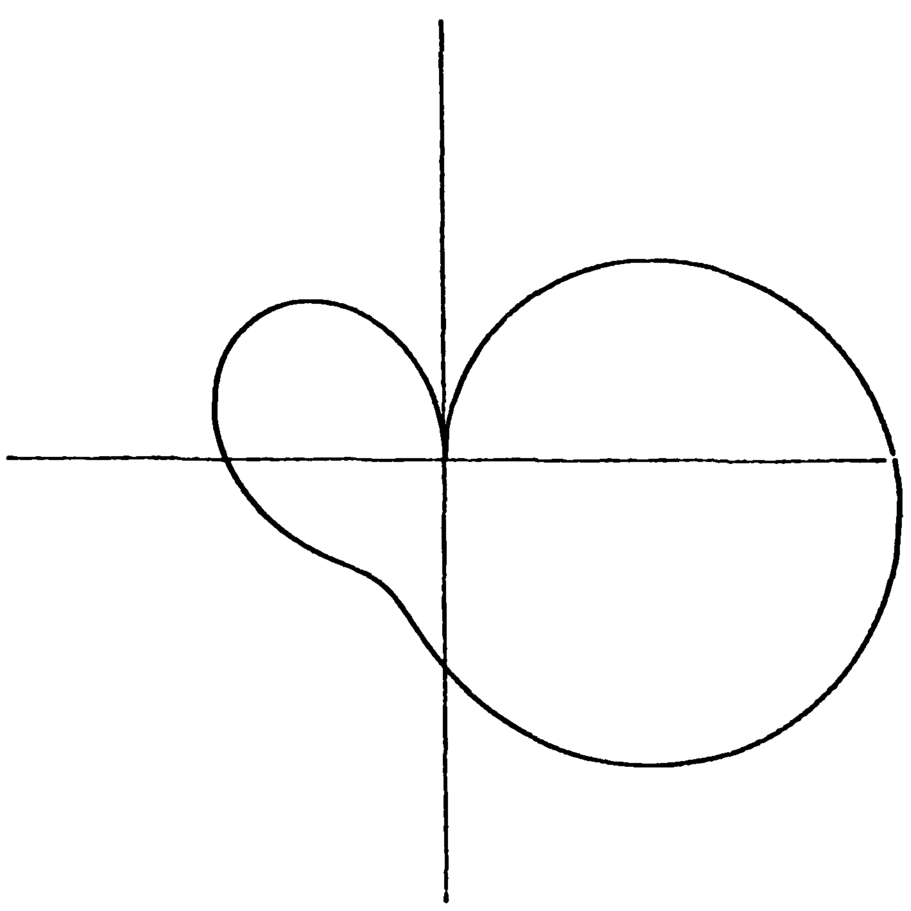
Figure 24 shows the radiation patten of the equilateral array when $\Theta_m = 0$ and $\Theta_n = 90$. The current distributions corresponding to this pattern is used as input into this example of the linearization design method. The initial guesses for the variables $V_{2Co}$, $V_{1Bo}$, $X_{C12o}$, and $X_{C23o}$ were

$$V_{2Co} = 1.0 + j1.0 \qquad V_{1Bo} = 1.0 + j1.0$$
$$X_{C23o} = 1.0 + j1.0 \qquad X_{C12o} = 1.0 + j1.0$$

The magnitude and phase of the currents flowing in each element of the array, the seven constant PI network parameters input by the user, and the resulting two designed parameters $X_{C12}$ and $X_{C23}$ are given below:

Magnitude and Phase of Desired Current Distributions

$$M_{I1} = 1.0000 \qquad PH_{I1} = 000.0000$$
$$M_{I2} = -0.5600 \qquad PH_{I2} = 120.0000$$
$$M_{I3} = 0.45 \qquad PH_{I3} = 023.000$$



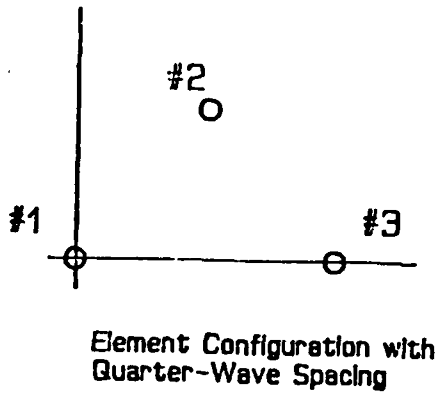


FIGURE 24 - Radiation Pattern Desired

Given PI Network Parameters

| | | |
|---|---|---|
| $X_{L13}$ = 10.0 | $X_{C13}$ = -1.0 | $X_{C31}$ = -5.0 |
| $X_{L12}$ = 10.0 | $X_{C21}$ = -1.0 | $X_{C12}$ = Unknown |
| $X_{L23}$ = 10.0 | $X_{C32}$ = -5.0 | $X_{C23}$ = Unknown |

Unknown Parameters for Ten Iterations

| | | |
|---|---|---|
| I = 1 | $X_{C12}$ = (3456.3+j234.6) | $X_{C23}$ = (00234.3-j485.2) |
| I = 2 | $X_{C12}$ = (0008.6-j005.1) | $X_{C23}$ = (-0.0001+j.1098) |
| I = 3 | $X_{C12}$ = (-.0018+j0.114) | $X_{C23}$ = (-0.0001-j.1099) |
| I = 4 | $X_{C12}$ = (0.0-j.1103) | $X_{C23}$ = (-0.0001-j.1099) |
| I = 5 | $X_{C12}$ = (0.0-j.1103) | $X_{C23}$ = (-0.0001-j.1099) |
| I = 6 | $X_{C12}$ = (0.0-j.1103) | $X_{C23}$ = (-0.0001-j.1099) |
| I = 7 | $X_{C12}$ = (0.0-j.1103) | $X_{C23}$ = (-0.0001-j.1099) |
| I = 8 | $X_{C12}$ = (0.0-j.1103) | $X_{C23}$ = (-0.0001-j.1099) |

The design program was run on a wide range of current distributions. The following results are the designed $X_{C12}$ and $X_{C23}$ and corresponding Q's for $\theta_m$ and $\theta_n$ values ranging from $\theta_m$ = 0 to 360 in steps of 90 degress and $\theta_n$ = 0 to 360 in steps of 45 degrees. Several of these runs are showns below with specific $\theta$m and $\theta$n values, corresponding current magnitudes and phases, the designed PI network parameters and corresponding Q values.

$\theta_m = 0.0$ $\theta_n = 45.0$

| | | | |
|---|---|---|---|
| $M_{I1}$ = | 1.00 | $PH_{I1}$ = | 21.0 |
| $M_{I2}$ = | 4.24 | $PH_{I2}$ = | 11.0 |
| $M_{I3}$ = | -4.57 | $PH_{I3}$ = | 22.0 |

| | | | |
|---|---|---|---|
| $X_{C13}$ = | (3.50E-7 + j0.1103) | $Q_{C13}$ = | 314658.1 |
| $X_{C23}$ = | (9.64E-7 + j0.1105) | $Q_{C23}$ = | 1141.9 |

$\theta_m = 0.0$ $\theta_n = 225.0$

| | | | |
|---|---|---|---|
| $M_{I1}$ = | 1.00 | $PH_{I1}$ = | -3.0 |
| $M_{I2}$ = | 0.47 | $PH_{I2}$ = | -23.0 |
| $M_{I3}$ = | -0.97 | $PH_{I3}$ = | 33.0 |

| | | | |
|---|---|---|---|
| $X_{C13}$ = | (2.15E-6 + j0.1103) | $Q_{C13}$ = | 51158.1 |
| $X_{C23}$ = | (5.18E-5 + j0.1104) | $Q_{C23}$ = | 21321.9 |

$\theta_m = 90.0$ $\theta_n = 135.0$

| | | | |
|---|---|---|---|
| $M_{I1}$ = | 1.00 | $PH_{I1}$ = | 7.0 |
| $M_{I2}$ = | -0.71 | $PH_{I2}$ = | -3.0 |
| $M_{I3}$ = | -0.35 | $PH_{I3}$ = | 43.0 |

$X_{C13}$ = (8.77E-6 + j0.1103)  $Q_{C13}$ = 12574.0

$X_{C23}$ = (-4.02E-4 + j0.1105)  $Q_{C23}$ = 275.9

From the above example it is obvious that the required capacitative reactances are complex. Therefore the Q's of the designed capacitors are non-infinite.

In order to check the results of this example, the given and calculated PI network parameters are input into the analysis program of Chapter Four. The current distributions resulting from the analysis subroutine are used to plot the radiation pattern. As indicated in Figure 25 the pattern produced by the analysis run is exactly the same of the pattern desired (Figure 24). However, if the resistive components of $X_{C12}$ and $X_{C23}$ are assumed zero, the currents produced by the analysis subroutine no longer result in the exact same pattern (see Figure 26). The figure shows that the null at $\Theta_n = 90$ is not as deep as it was previously.

## D. Conclusions

Surprisingly (given the complexity of the physical situation under discussion), the method usually converges. As seen in the above examples, if it is going to do so, it

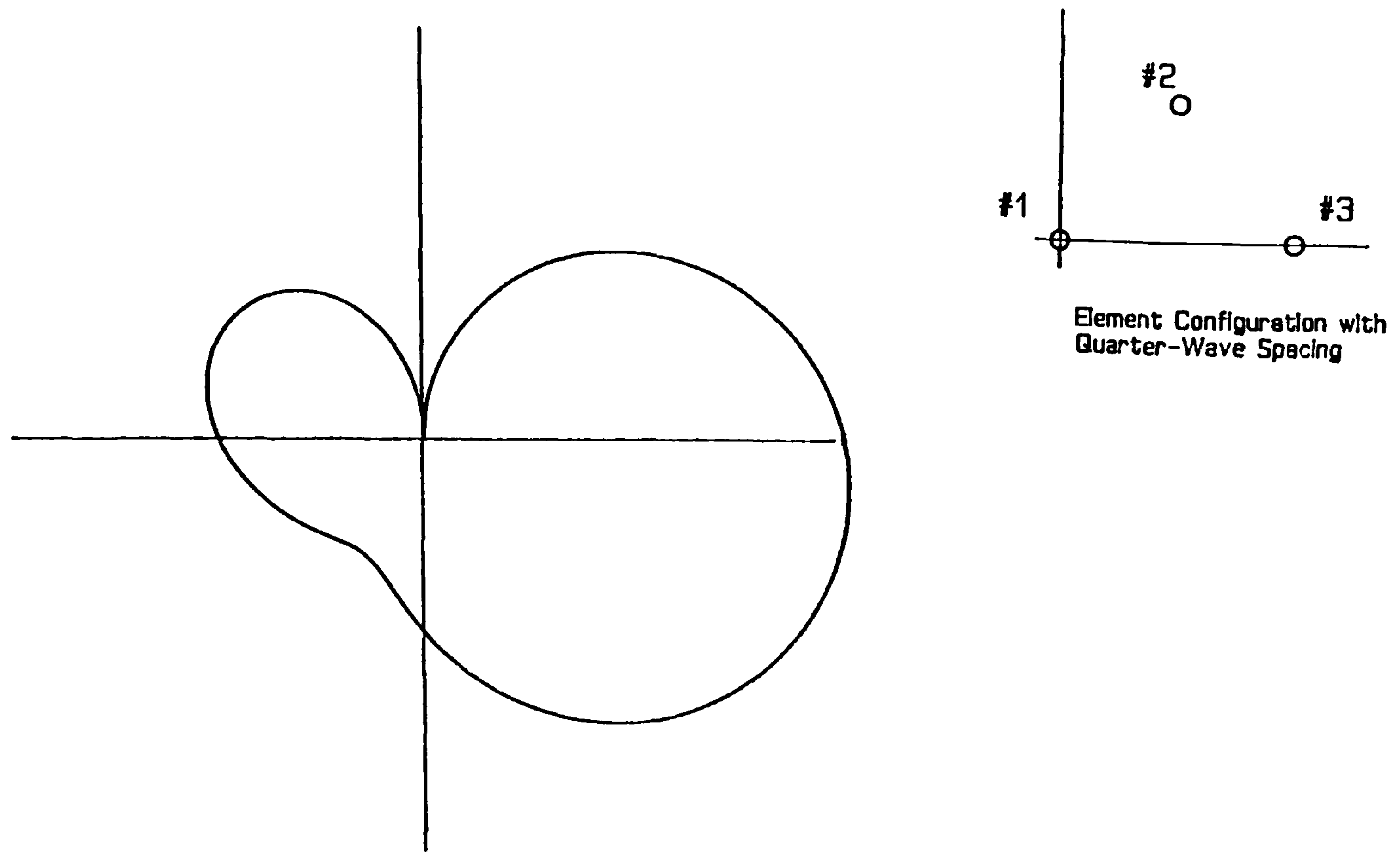


FIGURE 25 - Pattern from Designed Parameters

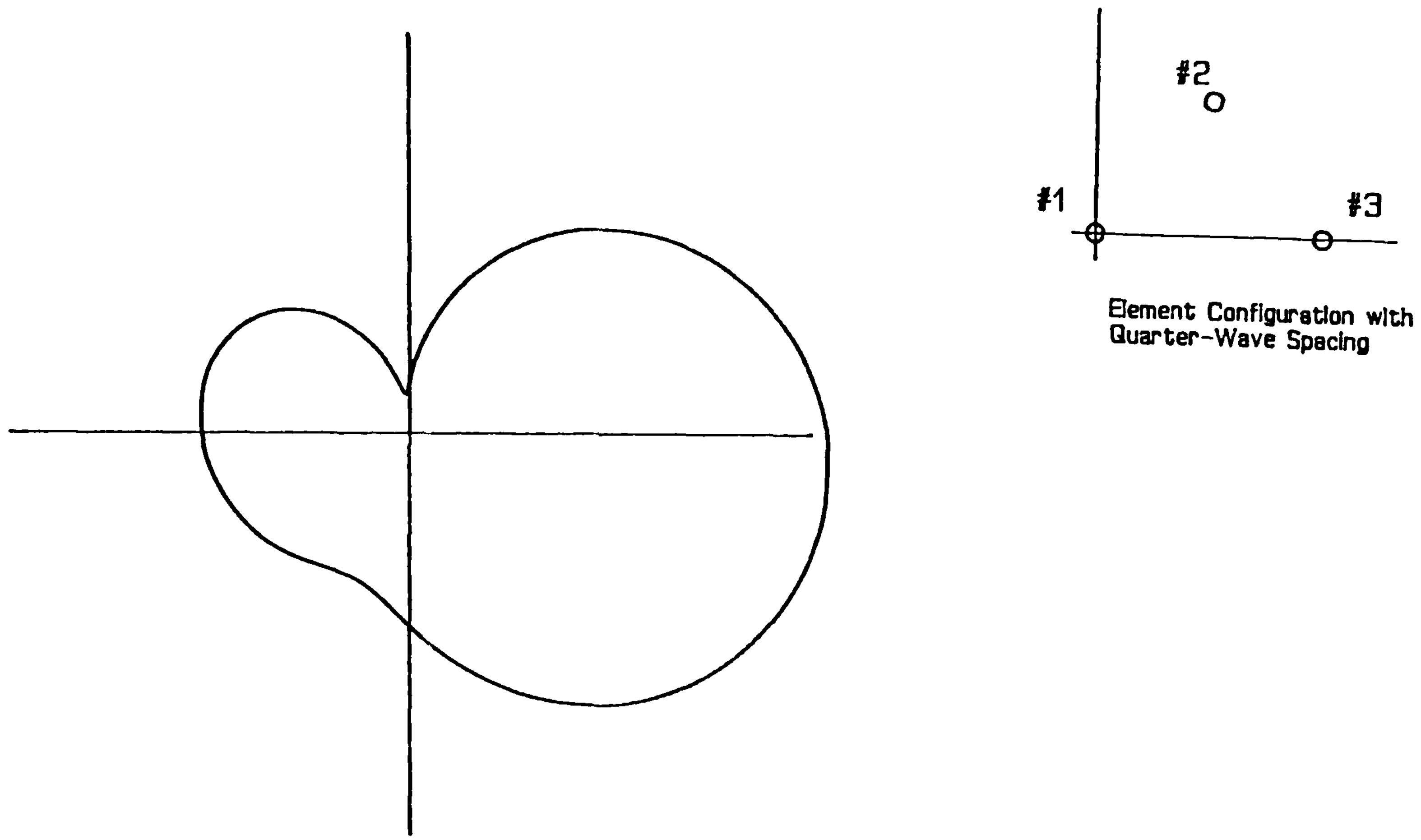


FIGURE 26 - Pattern with Zero Resistance
In the Designed Parameters

converges to single precision accuracy in about four iterations, yielding virtually instantaneous response on a multiuser VAX 11/780.

The main objective of th's chapter was to supply the reader with the essential components of the extended Newton's Method linearization design method for solving for the parameters of the PI interconnection networks. The 14 design equations were derived from the 24 complex equations describing the array and their source of nonlinearity was clearly indicated. The linearization process, which resulted in an iterative Newton's Method type procedure, was examined quite closely. This method converges for a majority of the patterns it is ask to produce.

This design procedure's biggest draw back is the fact that it only designs two of the nine PI network parameters. Also, if the given parameters are specified with magnitudes larger than 10.0 the design routine will not converge. However, the fact that this method works for any PI network parameters is in itself an achievement. This procedure has proven (at least in theory) that it is possible to use the interconnection networks of this thesis in a phased array antenna system.

# VI. A Direct Search Optimization Method: Developement and Testing

This chapter presents a detailed examination of a unique direct search optimization method developed by the author. The developement of this direct search optimization method was undertaken as a possible alternative technique for solving the nonlinear "design" equations of the equilateral array described in Chapter Five. There is definite application of this method in a proposed design technique based on the constrained optimization of the current distribution on the array's elements by adjustment of the PI network parameters in the "analysis" equations of Chapter Four. This design technique will be discussed in Chapter Seven.

A brief discussion of the generalities of direct search optimization methods begins this chapter. A detailed explanation of the various blocks of decision control for the direct search optimization method developed here is presented next. The operation of each of the separate procedures that are performed on the variables of the objective function are clearly explained. As an example, the various procedures of the optimization method are discussed in reference to a two variable quadratic equation. This chapter concludes with sample optimization trials on several quadratic and transcendental multivariable objective functions.

## A. General Operation of the Hillclimber

Direct search methods ( hillclimbing methods) of optimization base their operation on a finite number of evaluations of trial solutions to an objective function called the hill height. Comparision of subsequent trial solutions is the basis for further evaluation of the particular problem. The methods involve evaulations of the objective function starting with a given initial point in the particular n-dimensional space. This initial point is some arbitrary value of the variable vector $\underline{x} = (x_1, \ldots, x_n)^T$ containing the arguments of the function under consideration. These optimization methods usually only require objective function evaluation and do not use partial derivatives (as in the steepest decent optimization methods). Information accumulated as the search proceeds is used by some techniques in support of directional movements in the variables.

A two variable quadratic function of the form

$$f(x_1,x_2) = (x_1 - a)^2 + (x_2 - b)^2 \qquad (6.1)$$

will be used to help explain in detail the direct search optimization method to be discussed in this chapter. This optimization method will be referred to as the hillclimber throughout the following discussions. $x_1$ and $x_2$ define the axis of the two dimensional $x_1x_2$ plane. The a and b represent the location of the minimum of the simple quadratic function given in equation 6.1. Figure 27 shows the $x_1x_2$ plane and the level curves of the function represented by equation 6.1.

The hillclimber's basic operation, in relation to equation 6.1, is the adjustment of the variables $x_1$ and $x_2$ (starting from some initial point $(x_{1o},x_{2o})$ in the plane until the minimum of the objective function is realized. The variables are adjusted one at a time, in alternating sequence, with the magnitude of the objective function being calculated and evaluated after each adjustment of a variable (see Figure 28). The magnitude of the objective function is defined as the hill height of the function. For the two variable function of equation 6.1, with the minimum defined at a=1 and b=2 at an initial location $x_{1o}$=5 and $x_{2o}$=5, the hill height equals

$$\begin{aligned} \text{HH (Hill Height)} &= (5-1)^2 + (5-2)^2 \\ &= 25 \end{aligned}$$

The hillclimber adjusts the variables of a

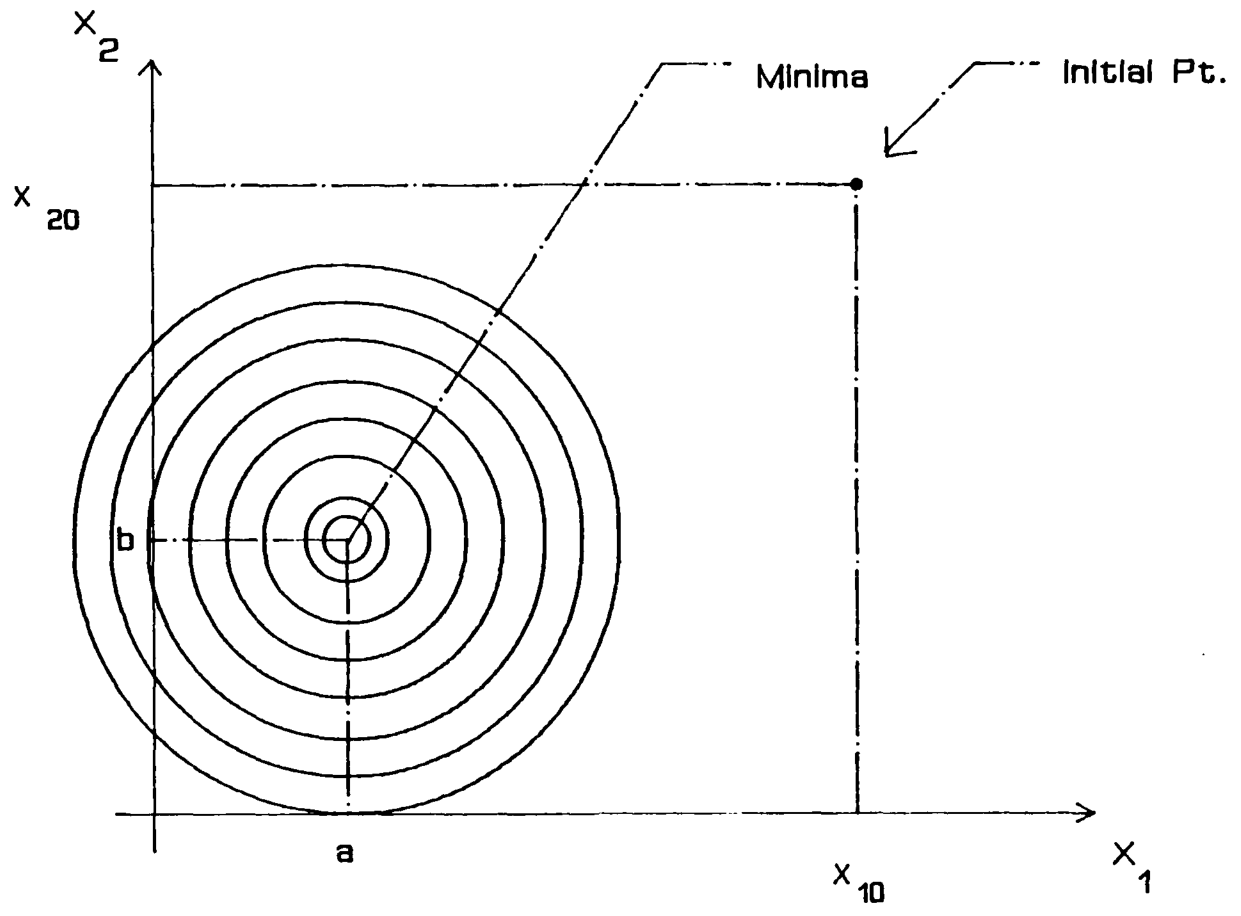


Figure 27 - Level Curves of Equation 6.1

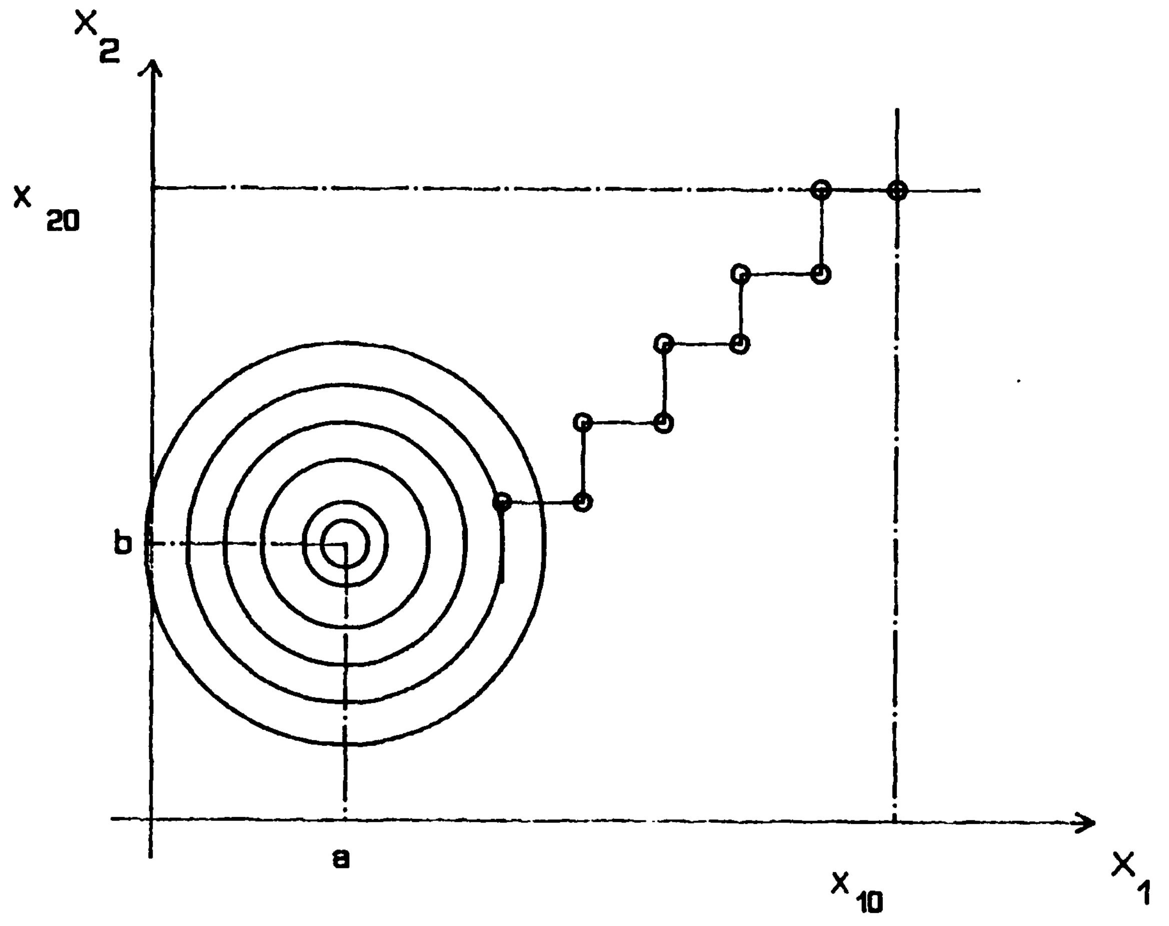


Figure 28 - Sequential Movement of Arguments X1 and X2

function in the following manner. The first variable of a multivariable sequence ( $x_1$, ...., $x_n$ ) is moved in a positive direction equal in magnitude to a present step size $s_i$. In other words, the initial given value $x_{1o}$ for the variable $x_1$ is replaced by $x_{1o} + s_1 * d_1$ where $s_1$ is the step size and $d_1$ is the direction of the variable $x_1$; $d_1$ is either +1 or -1. The hillclimber has moved the location of the variables of the two dimensional function of equation 6.1 from $(x_{1o}, x_{2o})$ to the new location $(x_{1o} + s_1 * d_1, x_{2o})$. The hill height of the function is calculated, and by the comparision of the new hill height $HH_1$ to the previous hill height $HH_0$ (the hill height calculated from the initial values of the variables) the hillclimber routine is able to make basic adjustment decisions as to new values for $x_1$ and $x_2$ - hopefully closer to the new location of the minimum.

At some point in the optimization process (sometime during the first adjustment sequence through all of the variables of the objective function) the hill height becomes the hill height of least magnitude to date. The hill height of least magnitude becomes the previous hill height upon which the basic adjustment decisions are made. This previous hill height will be referred to as the lowest hill height to date ($HH_{ltd}$).

## B. Hillclimber's Fundamental Control Procedures

Comparision of the present and the lowest hill heights to date logically can result in three possible situations as a result of a change in one of the $x_i$: the hill height could have increased ($HH_1 > HH_{1td}$), it could have decreased ($HH_1 < HH_{1td}$), or it could have stayed the same ($HH_1 = HH_{1td}$). What the hillclimber does in each of the above situations will be discussed next.

First, let us assume a new hill height has been calculated and the magnitude is not lower than the least hill height to date ($HH_1 > HH_{1td}$). The decisions of the hillclimber are as follows: since the value of the hill height is now greater than it previously was (because it moved further away from the desired value), the variable $x_1$ shall be returned to the previous value $x_{1o}$ and the direction $d_1$ shall be changed to the direction opposite to what it was previously. These two decisions constitute the first fundemental block of decision control code and are shown more clearly in the notation below:

```
IF ($HH_1 > HH_{1td}$) THEN
    1.  $x_i = x_i - s_i*d_i$ (returned to $x_i$)
    2.  $d_i = -1*d_i$
```

When the hillheight $H_i$ is less than the least hill height to date $HH_{1td}$ ($HH_1 < HH_{1td}$), the variable $x_1$ is allowed to retain the new value ($x_1+s_1*d_1$) and current direction while the succeeding variable $x_2$ is immediately adjusted to the new value $x_2+s_2*d_2$. This is shown below in notation form as

```
IF ( $HH_1 < HH_{1td}$ ) THEN
   1. $x_i = x_i + s_i*d_i$
   2. $x_{(i+1)} = x_{(i+1)} + s_{(i+1)}*d_{(i+1)}$
```

In situations where the present hill height equals the current least hill height ($HH_1 = HH_{1td}$) the variable $x_1$ is returned to the previous value and the present direction is maintained. The succeeding variable $x_2$ is immediately adjusted using the current step size and direction. In notational form these decisions constitute the third fundemental block of decision control code and are

```
IF ( $HH_1 = HH_{1td}$) THEN
   1. $x_i = x_i - s_i*d_i$
   2. $x_{(i+1)} = x_{(i+1)o} + s_{(i+1)}*d_{(i+1)}$
   3. $s_i = s_i*10.0$
```

C. <u>Hill Height Not Changing ($HH_{i-1} = HH_i$)</u>

Another block of fundamental decision code is used to handle situations where adjustment of the variables of the objective function does not change the magnitude of the hill height. In other words the hill height is not being changed within the word length of the computer making the calculations. This can happen when the variables being adjusted have reached values that are too large or too small to have any effect on the objective function, or whenever the surface is flat. This problem is detected by continually comparing each new hill height (there will be one hill height for each iteration of the process) with the hill height immediately previous.

An example of the operation of this block of decision code is shown by defining (I) to be the iteration counter of the optimization process. I=0 corresponds to the hill height calculated from the given initial values for the variables. In a two variable optimizaton process, I=7 corresponds to the fourth adjustment of the first variable $x_1$. Therefore, $HH_o$ is compared to $HH_1$ and for the remaining iterations $HH_{(i-1)}$ is compared to $HH_i$ inorder to detect when the adjustment of the variables is not effecting the magnitude of the hill height. If this condition is detected the following action is initiated:

```
IF ($HH_{(i-1)} = HH_i$) THEN
    1. $x_i = x_i - s_i*d_i$ (returned to $x_i$)
```

2. $x_{(i+1)} = x_{(i+1)} + s_{(i+1)}*d_{(i+1)}$
3. $s_i = s_i*10.0$

Now that the four fundamental blocks of variable adjustment decisions have been discussed, the two variable optimization process referred to in that discussion will be continued in some what more detail.

Using these four fundamental decision blocks the two variables of equation 6.1 will continually be moved towards the desired values ($x_1$=a, $x_2$=b), resulting in a hill height of zero magnitude (see Figure 28). The optimization process will continue the sequence of alternating adjustments until one of the variables makes a move in a direction that is further away from the location of the minimum ($HH_i > HH_{1td}$). Then the hillclimber reverses the variable's direction $d_i$ as shown in the preceeding paragraphs.

## D. Periodic Pattern Detection

Figure 29 shows this happening to the $x_2$ variable as it moves below the minimum b to an increased hill height ($HH_i > HH_{1td}$). At this point, the $x_2$ variable will be moved back to the value it had just prior to the step that increased the hill height and the direction $d_1$

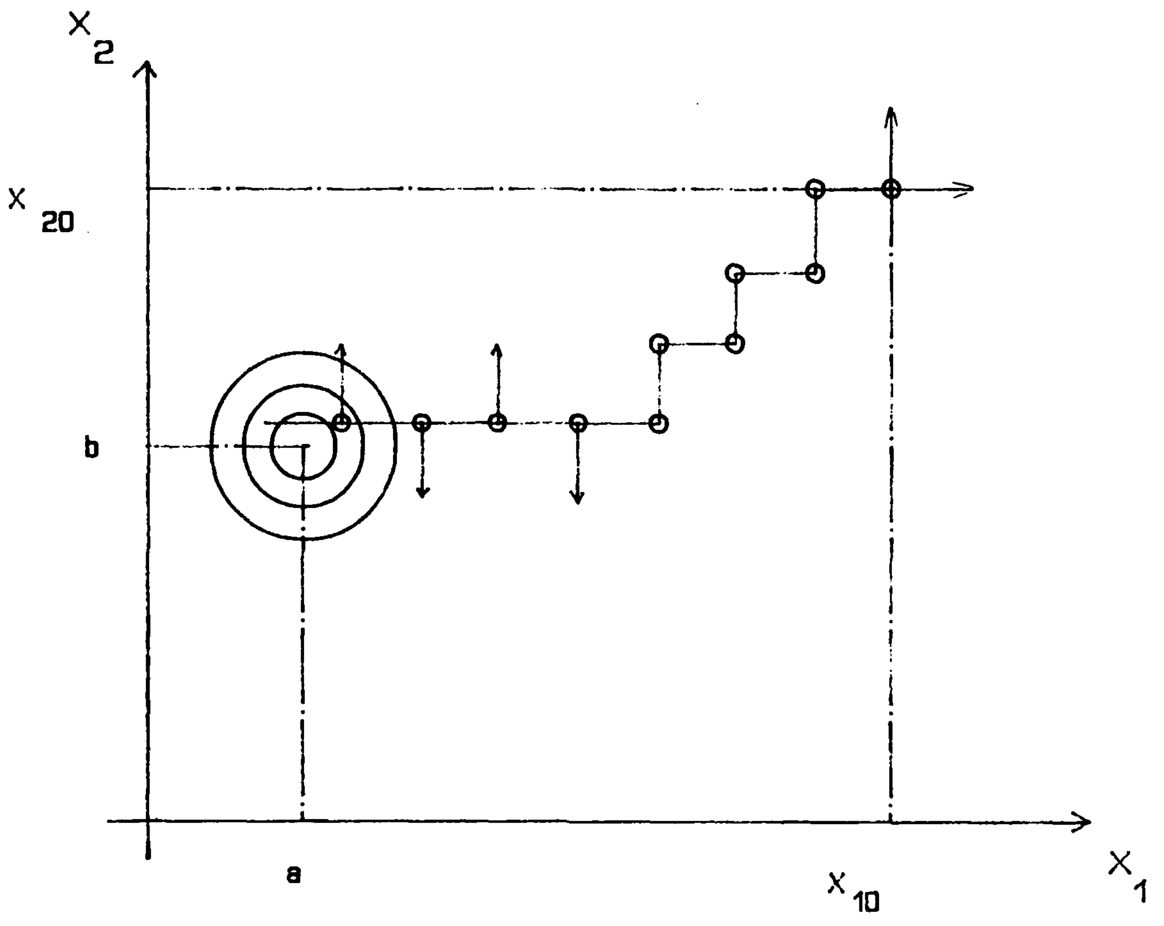


Figure 29 -- Periodic Behavior in the X2 Argument

is multiplied by -1. $x_1$ will continue to be moved in the same direction because it still can move much closer to the minimum coordinate (a) by maintaining this direction. The $x_2$ variable will now step in the direction $d_2$ and from Figure 29 one can see that this step will also be retracted because the hill height was not reduced below the lowest to date ($HH_{1td}$). <u>The $x_2$ variable will now begin a sustained sequence of back and forth steps on each side of the hill height of least magnitude to date.</u> This back and forth sequence will consist of three $x_2$ variable magnitudes; the $x_2$ magnitude corresponding to the lowest hill height ($HH_{1td}$), an $x_2$ magnitude corresponding to a hill height lower in magnitude than $HH_{1td}$, and an $x_2$ magnitude corresponding to a hill height greater in magnitude than the $HH_{1td}$. Detection of this periodic sequence of steps above and below the present best variable magnitude constitutes the most important block of decision control within this optimization routine: <u>the reduction of step sizes upon detection of oscillatory behavior in the adjustment of the variable's directions and magnitudes.</u>

The periodic behavior described in the above paragraph happens because the size of the steps being taken in the positive and negative directions are too large. For instance, if the current best magnitude found for $x_2$ was 50, the desired or minima value (b) equals

54.3, and the step size $s_2$ was equal to 10 then the routine would continually jump from 50 to 60, from 60 back to 50, and then from 50 to 40 and back to 50 again. With an $s_2$ of magnitude 10 maintained, the routine would forever repeat this pattern. This periodic pattern of movements is shown in Figure 30 for both $x_1$ and $x_2$ arguments. However, the routine is able to detect this periodic behavior in both arguments and adjust the step size accordingly.

For the two variable example being discussed, the step size $s_2$ will be decreased from its initial value of 10 to a value of 1.0 and the routine will begin to move from the best magnitude 50 in the positive direction toward 54.3. But, once the $x_2$ variable reaches the magnitude of 54, a periodic movement above and below this new best magnitude for $x_2$ will begin again. $s_2$ will again be reduced by a factor of 10 to equal 0.1. The routine will continue movement towards the minimum and will quickly find the desired value since the routine is taking steps in the decimal limits of the minimum (0.1). This periodic behavior will happen on the $x_1$ variable as it nears the minima (a) and $s_1$ will also be decreased to the decimal accuracy of the minimum.

The periodic pattern detection procedure uses an array containing the previous values of the variables choosen by the hillclimber during the progression of the

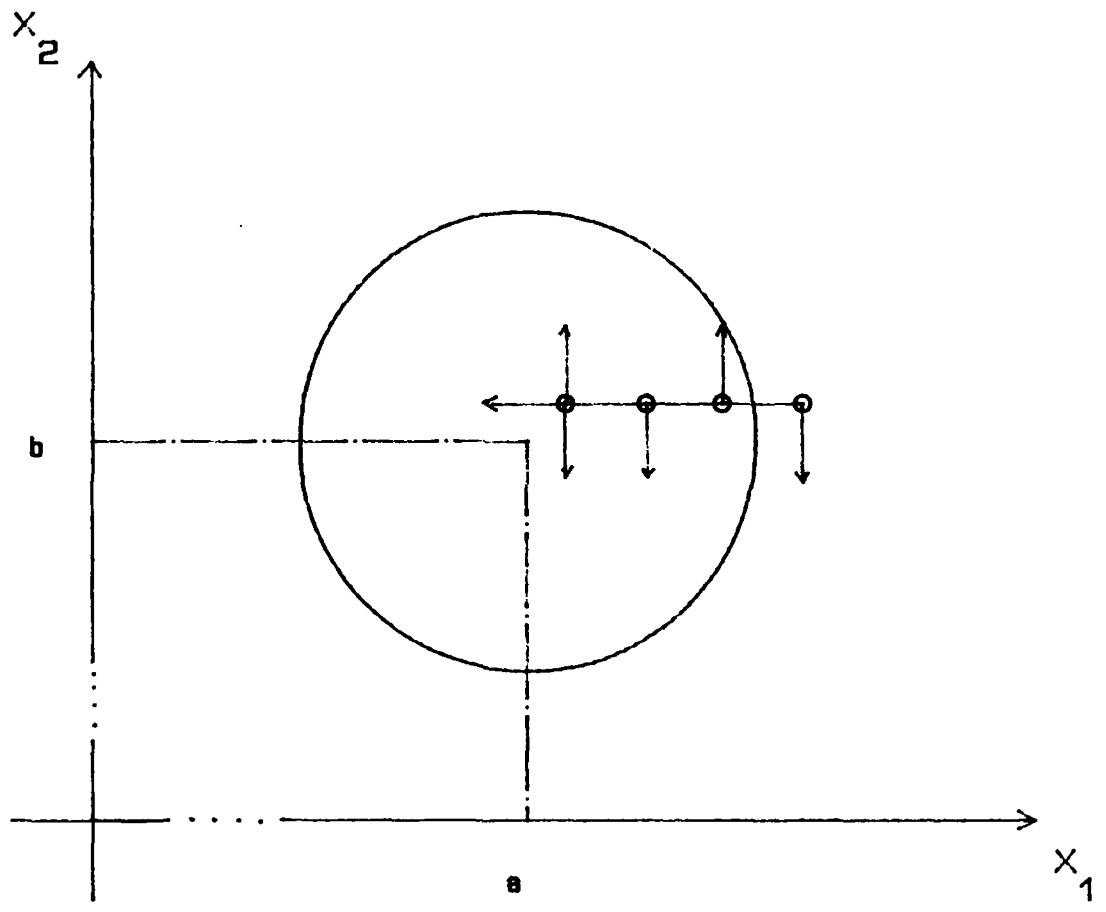


Figure 30 - Periodic Behavior in both X1 and X2 Arguments

routine. The length of a periodic pattern is twice the number of variables. If (I) is defined as the iteration counter of the optimization process then the periodic pattern detector compare~ the 2xN (where N is the number of arguments in $f(x_i)$) sequential values $x_{i(I-7)}$ thru $x_{i(I-4)}$ of a variable with the next 2xN set of sequential values $x_{i(I-3)}$ thru $x_{i(I)}$ after each iteration of the optimization routine. If there are two variables in the function being optimized a periodic pattern four iterations in length can occur. Detection is accomplished by comparing the the present value of the variable and its previous seven values. If $x_1$ is the variable being observed the comparisions are as follows:

| | | | |
|---|---|---|---|
| Does | $X_{1(I-0)}$ | equal | $X_{1(I-4)}$ |
| Does | $X_{1(I-1)}$ | equal | $X_{1(I-5)}$ |
| Does | $X_{1(I-2)}$ | equal | $X_{1(I-6)}$ |
| Does | $X_{1(I-3)}$ | equal | $X_{1(I-7)}$ |

If the four statements above are true then

1. $s_i = s_i/10.0$ (step size is reduced)
2. $x_i = x_{1hhi}$ ($x_i$ is set to best value)

where $x_{1hhi}$ equals the magnitude of the variable corresponding to the lowest hill height found to date.

If all four comparisons are equal, a periodic pattern has been detected and $x_1$ is oscillating around the best value it can find with the present $s_i$ magnitude.

## E. Prevention of False Pattern Detection

Because the variables are reset to the magnitude corresponding to the lowest hill height found so far in the process it is possible that the pattern detector could, depending on the present magnitude or even the minimum of the function, detect another periodic pattern on the next adjustment of a variable $x_{(i+1)}$. This pattern would be false because the routine had not been given the chance to adjust the particular variable using the reduced step size. Therefore a new periodic pattern could not exist one iteration past the detection of a previous pattern. Preventing detection until 2n+1 iterations beyond the detection of a previous pattern insures the variable step sizes do not get reduced on every iteration.

## F. Step Size Enlargement

Reduction of the step size magnitude is imperative to finding the actual minimum of a function. On the other hand step size enlargements, while not an absolute necessity of the optimization routine, do increase the speed of convergence by allowing the variables to move in larger steps towards the desired results. The step size enlargement module depends on a user input constant Mdist, which specifies the length of continuous movement in the same direction. The word "length" means the number of movements of a particular variable in the same direction. The Mdist variable should usually be specified to be 10 steps in the same direction, but has been given other values in test runs to see if there is any optimal length the variables should be allowed to move in the same direction before the step size is enlarged. For the objective functions tested, values of Mdist anywhere between 5 and 13 seem to perform equally well. A length of 10 was chosen for all the examples in this thesis because all other step size evaluation procedures base their movements, reductions, and comparisons on the factor of 10. This length variable can be input to a users main program and passed into the optimization routine for the purpose of tuning the hillclimber to the users particular problem.

## G. Step Size Boundaries

There are two procedures in this direct search optimization routine which prevent the step size of the variables from becoming too small or from becoming too large. These procedures are refered to as the step size maximum and minimum. The user is able to specify the step size range for all variables and the step sizes are prevented from going beyond or below these limits. The step size maximum procedure checks the step sizes against the maximum magnitude allowed and resets them to a magnitude of ten below the limit specified by the user. Immediately after this step size maximum procedure, the step size minimum module checks the step size of all the variables against the minimum allowed. If one of the variable's step sizes is at the minimum magnitude it will be reset to a value that is a factor ten above the limit and the particular variable in question is reset to the best value found (to date). The step size can be continually increased if the variable continues to move in one direction. This can happen if the movements of the variable have only a very small effect on the hill height. The step size maximum procedure prevents this from happening.

## H. Basic Procedures Reviewed

There are five distinct control modules for this optimization routine and they are as follows:

1. Direction Reversal (due to increase in hill height)
2. Step Size Reduction (due to periodic pattern detection)
3. Step Size Enlargement (to increase convergence rate)
4. Step Size Maximum Limit (to prevent divergence)
5. Step Size Minimum Limit (to prevent divergence)

This optimization routine is in the form of a subroutine which is called by a main program. The main program is nothing more than a medium for inputing the initial values $x_{io}$ and step sizes $s_{io}$ and transferring them to the hillclimber. The hillclimber will call a user created subroutine containing the function to be optimized. The hillclimber passes the current values of the variables to the subroutine FUNC where the hill height is computed. Appendix E contains a listing of the hillclimber subroutine HCSUBR, the objective function subroutine FUNC, and the main program MAINHC.

## I. Results

The hillclimber was tested on several quadratic and transcendental functions of which a two will be shown below. The examples will present the form of the objective function, the initial values of the variables and their step sizes, the step size maximum and minimum limits, along with the $HH_{1td}$ and the corresponding optimized values of the variables.

The first function tested has the form,

$$F(x_1) = (x_1{}^2 - 78.7)^2$$

with the hill height equal to

$$HH_i = (x_1{}^2 - 78.7)^2$$

The initial values and other parameters of interest are

$$x_{1o} = 10.0 \qquad s_{max} = 1000.0$$
$$s_{1o} = 1.0 \qquad s_{min} = 0.0001$$

and the results after 67 iterations are

$$x_{11hh} = 8.8713 \qquad HH_{1td} = 0.00$$

The table of data shown below contains the value for the argument $x_1$ and the resulting hill height for the first 20 iterations of the hillclimber on this first example.

| I | $x_1$ | $F(x_1)$ |
|---|---|---|
| 1 | 10.0 | 453.69 |
| 2 | 11.0 | 1789.29 |
| 3 | 10.0 | 453.69 |
| 4 | 9.0 | 5.29 |
| 5 | 9.0 | 5.29 |
| 6 | 8.0 | 216.09 |
| 7 | 9.0 | 5.29 |
| 8 | 10.0 | 453.69 |
| 9 | 9.0 | 5.29 |
| 10 | 8.0 | 216.09 |
| 11 | 9.0 | 5.29 |
| 12 | 10.0 | 453.69 |
| 13 | 8.9 | 0.26 |
| 14 | 8.8 | 1.59 |
| 15 | 8.9 | 0.26 |
| 16 | 9.0 | 5.29 |
| 17 | 8.9 | 0.26 |
| 18 | 8.8 | 1.59 |
| 19 | 8.9 | 0.26 |
| 20 | 9.0 | 5.29 |

The second function used to test this hillclimber is of the form

$$F(x1,x2) = [x_2^2 - 20\sin(0.05x_1)]^2 + 0.1[x_2^2 + x_1^2]$$

The initial values and other parameters of interest are

$x_{1o} = 100.0$ $x2o = 100.0$ $s_{max} = 1000.0$

$s_{1o} = 10.0$ $s2o = 10.0$ $s_{min} = 0.0001$

and the results after 200 iterations are

$x_{1hh} = 0.10$ $x_{2hh} = 0.11$ $HH_{1td} = 2.3E-2$

## J. Conclusions

For the relatively simple functions tested, this unique hillclimbing routine seems to work quite well. The initial guesses can usually be varied by the user until an optimal solution to the function or functions is reached. The step size enlargement procedure contains a discretionary variable Mdist. This variable can be specified by the user if desired and therefore creates some uncertainity in this procedure. The author used a value of ten for this variable. The four fundamental blocks of decision control code and the periodic pattern detector are the most important parts of this routine and the logic behind them is quite sound.

Since developement of this routine was only a part of my entire thesis, further testing and refinement of the algorithm is in order. This routine was use in the beam steering procedure of Chapter Two and the results were quite satisfactory. The radiation patterns always had the main lobe fairly close to the desired direction. The hillclimber is also implemented in a phased array design procedure to be discussed in Chapter Seven.

# VII. Interconnection Network Design: Direct Search Optimization of Constrained Analysis Equations

In Chapter Four quantitative analysis of the 24 complex equations describing the physical and electrical characteristics of the three element equilateral array was performed. The analysis of the equilateral array, given values for all of the PI network parameters, results in the current distribution on the array. The hillclimbing technique presented in Chapter Six was developed specifically for application in design of interconnection networks for the phased array system of Figure 5. In this chapter, the hillclimber will be used to adjust the PI network parameters until a desired current distribution is realized.

The discussion begins with an explanation of how the calculated and desired current distributions are defined. How the hill height is defined and calculated is discussed next. All of the procedures of this design method have been programmed in FORTRAN. An explanation of the structure and interaction of the various subroutines is presented. The subroutines of this design method are pulled together in a main program which accepts inputs and provides the resulting output. The output of this design method is the best current distribution found after a finite number of iterations of the hillclimber. Sample runs of this design method showing the resulting current distributions and PI network parameters are presented at the end of this chapter.

The design method of this chapter uses the hillclimber discussed in Chapter Six to adjust the parameters of the PI networks to whatever values will result in the desired currents flowing in the elements of the equilateral phased array of Figure 5. The quantitative analysis of this system of antennas has been placed in a subroutine CURR which is called by the hillclimber subroutine CAP6HC. The subroutine CURR returns the complex values of the current distributions on the elements for the PI network parameters provided by the hillclimber.

## A. Hill Height Defined

The current distributions desired by a user (corresponding to the desired pattern) are provided as input to this design method and are part of the hill height. These desired currents ($I_1$, $I_2$, and $I_3$) are normalized and used as the minima of the solution to the linear equations describing the array. The desired current distributions of elements #2 and #3 are normalized to the value that corresponds to a current distribution on element #1 of amplitude 1.0 and phase 0.0 and take the following form:

$$I_{1n} = \frac{I_1}{I_1} \qquad I_{2n} = \frac{I_2}{I_1} \qquad I_{3n} = \frac{I_3}{I_1}$$

The currents being calculated after each iteration of the hillclimber ($b_1$, $b_2$, and $b_3$) are also normalized to values with respect to the current distribution on element #1 and are shown below.

$$b_{1n} = \frac{b_1}{b_1} \qquad b_{2n} = \frac{b_2}{b_1} \qquad b_{3n} = \frac{b_3}{b_1}$$

The PI network parameters are loaded with the initial guess values ($x_{io}$) and passed into the analysis subroutine. After the 15 complex linear equations are solved, the analysis subroutine CURR passes the computed current distributions back into the main program where they are normalized as shown above. The hill height is defined to be the squared difference between the normalized desired currents ($I_{1n}$, $I_{2n}$, and $I_{3n}$) and the normalized calculated currents ($b_{1n}$, $b_{2n}$, and $b_{3n}$) and is shown in equation 7.1. The hill height is calculated after each adjustment of a PI network parameter during the progression of the optimization routine. The real and imaginary components of these complex current distributions are subtracted and the difference is then squared. The resulting squared differences between the currents real and imaginary components are then added together and the sum is the hill height upon which the hillclimber will base decisions. The hill height takes the following form:

$$\begin{aligned} HH = {} & [Re(I_{1n})-Re(b_{1n})]^2 + [Re(I_{2n})-Re(b_{2n})]^2 \\ & + [Re(I_{3n})-Re(b_{3n})]^2 + [Im(I_{1n})-Im(b_{1n})]^2 \\ & + [Im(I_{2n})-Im(b_{2n})]^2 + [Im(I_{3n})-Im(b_{3n})]^2 \qquad (7.1) \end{aligned}$$

Using input statements in the main program the variable vector $x_i$, the iteration count I, the number of variables in the objective function (Nvar), the variable's intial step sizes $s_i$ , and the range of the step sizes $s_{max}$ and $s_{min;}$ are all input by user. With these inputs from the main program the hillclimber is able to evaluate the hill height magnitude and use any periodic behavior in the variables in order to make the decisions necessary for adjustment of the variables.

Upon completion of the specified number of iterations the hillclimber returns the best calculated current distributions along with the values of the PI network parameters that produce these currents. These best calculated currents are normalized and used as inputs to a subroutine which will plot the resulting radiation pattern for the triangular geometry of the array. Also the hillclimber can be conveyed variables that allow the user to observe the pattern being produced by the best current distributions found to date, at whatever iteration count, during the hillclimbers progression. Listings of the program used in this design procedure are given in Appendix E.

## B. Results

With only minor changes in the programs listed in appendix F the PI network parameters can be adjusted in any combination and sequence. The programs listed in appendix F are designed for adjustment of the six capacitive reactances while the three inductive reactances are held constant. The results to be presented in this chapter will also include the problems of adjustment of the three inductive reactances with the six capacitive reactances held constant and adjustment of all nine PI network parameters. The specific data items to be listed for the three design problems being presented are the desired radiation pattern and corresponding current distribution, the reactances of the Pi network parameters held constant, the initial magnitude and step size of the parameters being adjusted by the hillclimber, the best current distribution resulting from a finite number of iterations of the hillclimber and corresponding designed PI network parameters, and the resulting radiation pattern.

The current distributions corresponding a radiation pattern with $\Theta_m = 60$ and $\Theta_n = 180$ will be used as input for the three design problems. This desired radiation pattern is shown in Figure 31 and can be used to compared with the radiation patterns produced by the three design problems to be presented below.

The first example uses the hillclimber to adjust the three inductive reactances of the three PI networks, given constant values for the six capacitative reactances, until the desired pattern is realized. The resulting current distributions and corresponding designed inductances are presented next and the radiation pattern

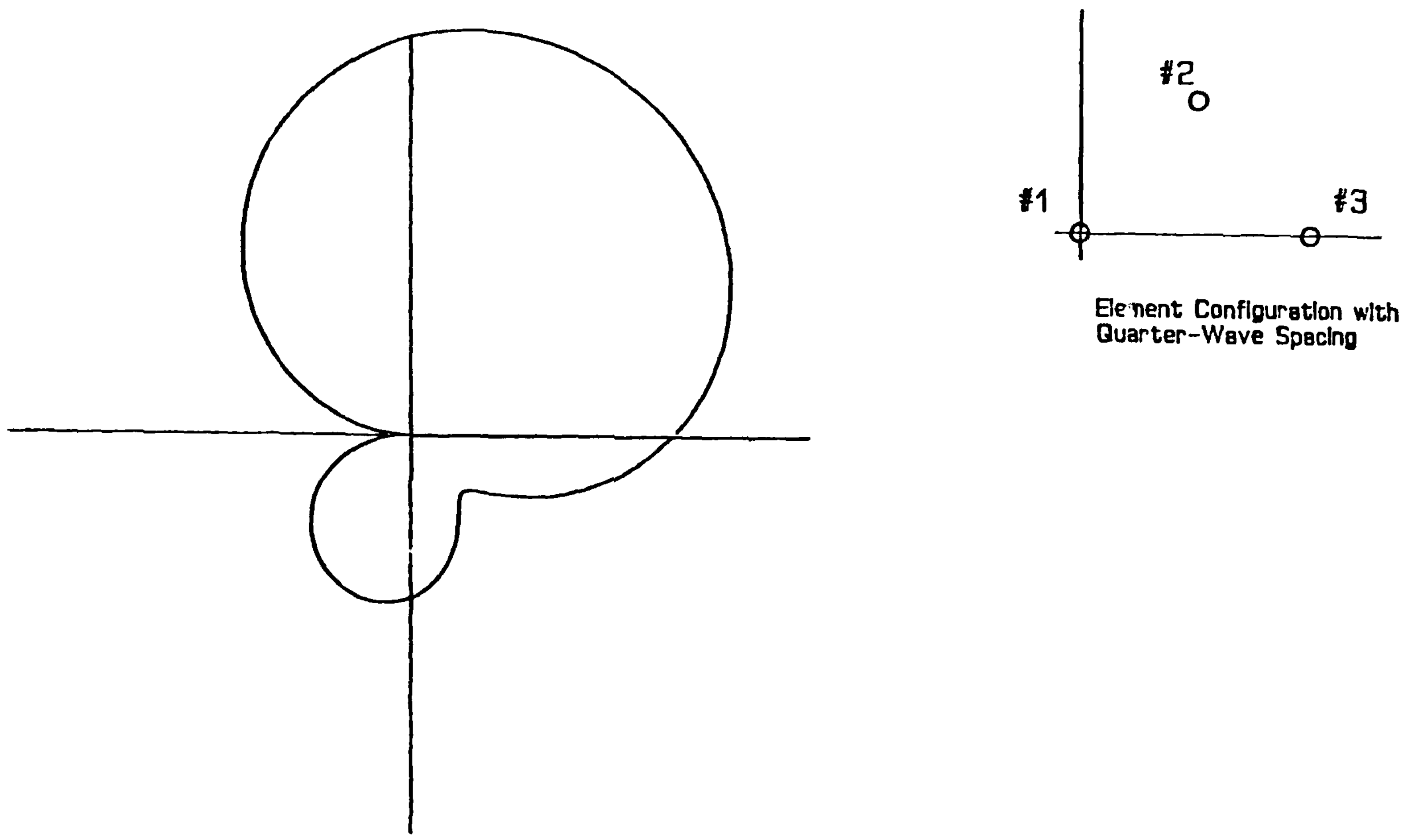


FIGURE 31 - Desired Pattern. $\theta m=60$ & $\theta n=180$

is shown in Figure 32.

Desired Current Distribution in Magnitude and Phase

| | |
|---|---|
| $M_{I1} = 1.000$ | $PH_{I1} = 0.00$ |
| $M_{I2} = 0.8480$ | $PH_{I2} = -148.05$ |
| $M_{I3} = 0.2593$ | $PH_{I3} = -42.15$ |

Given Capacitive Reactances

| | |
|---|---|
| $X_{C13} = -100$ | 227 pF |
| $X_{C31} = -200$ | 113 pF |
| $X_{C12} = -300$ | 75 pF |
| $X_{C21} = -400$ | 57 pF |
| $X_{C23} = -500$ | 45 pH |
| $X_{C32} = -600$ | 38 pF |

Initial Values and Step Sizes

| | |
|---|---|
| $X_{L13} = X_{1o} = 100.0$ | $S_{1o} = 10.0$ |
| $X_{L12} = X_3 = 134.0$ | $S_{3o} = 10.0$ |
| $X_{L23} = X_{5o} = 23.0$ | $S_{5o} = 1.0$ |

Hillclimber Parameters

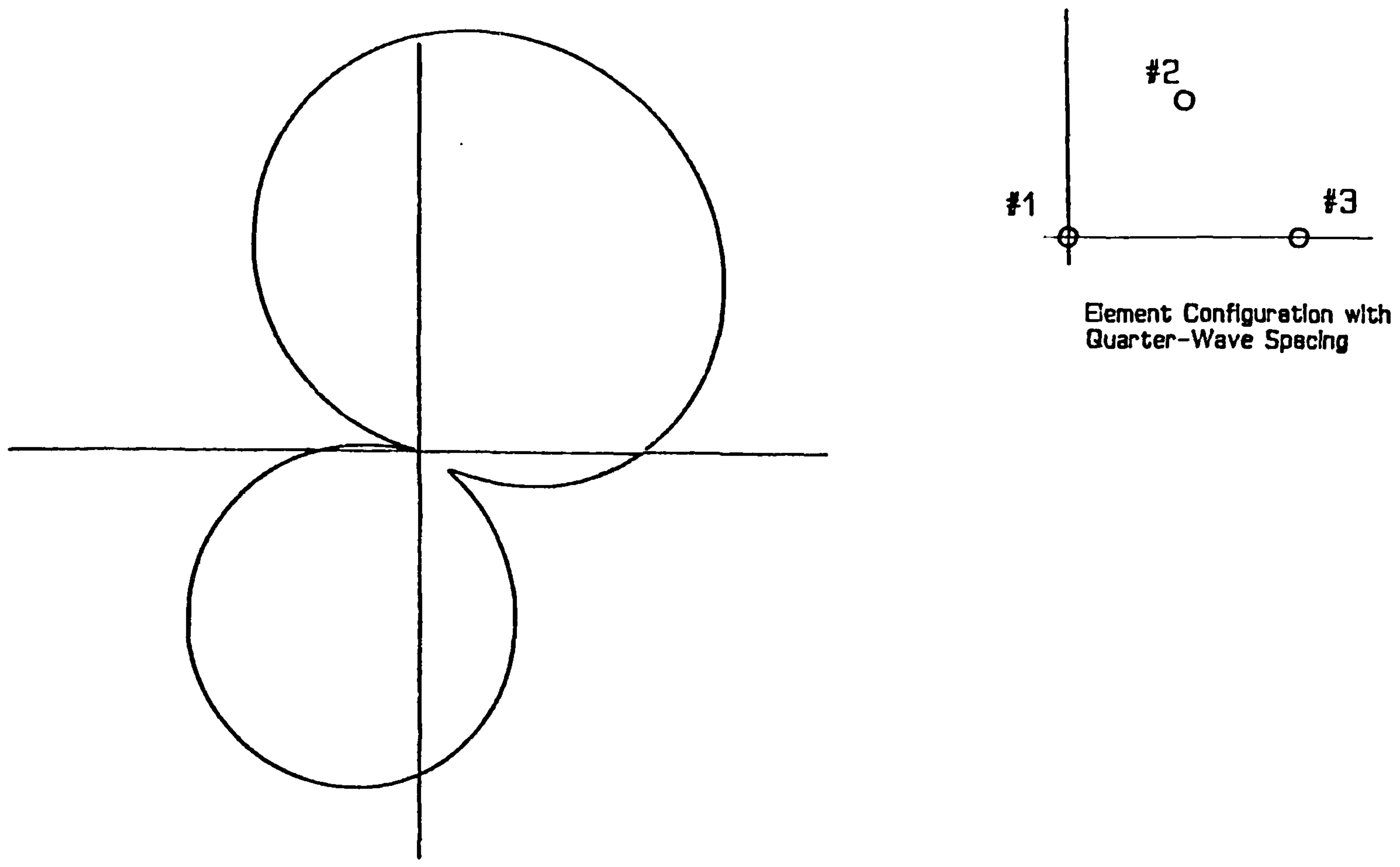


FIGURE 32 - Pattern from 3 Inductor Problem

$I = 300 \quad S_{max} = 10000.0 \quad S_{min} = 0.00001$

Initial Value of Hill Height

$HH_1 = 2.331952$

Calculated Currrent Distribution

$M_{b1n} = 1.0000 \quad PH_{b1n} = 0.000$

$M_{b2n} = 1.1239 \quad PH_{b2n} = -164.7$

$M_{b3n} = 0.2011 \quad PH_{b3n} = 23.08$

Resulting Inductive Reactances and Inductance at 7 MHz

$X_{L13} = 291925 \quad 6.64$ mH

$X_{L12} = -85.375 \quad 266$ pF

$X_{L23} = 1650515 \quad 37.5$ mH

The second design problem involves adjustment of the six capacitative reactances, while the inductances are held constant. After 500 iterations of the hillclimber the resulting radiation pattern is plotted and is shown in Figure 33. The following list of data is in the same format as the first design problem.

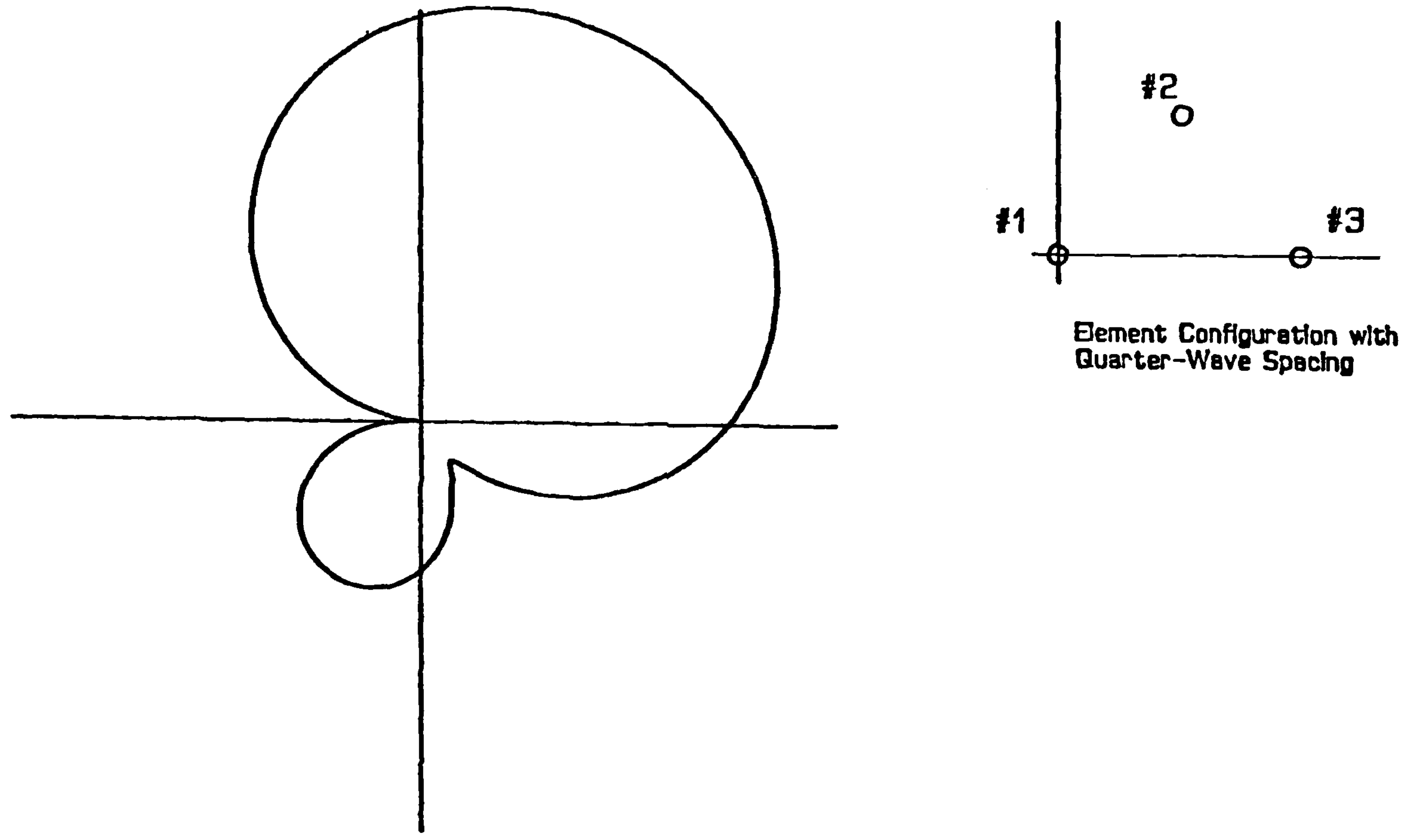


FIGURE 33 - Pattern from 6 Capacitor Problem

Desired Current Distribution in Magnitude and Phase

| | |
|---|---|
| $M_{I1} = 1.000$ | $PH_{I1} = 0.00$ |
| $M_{I2} = 0.8480$ | $PH_{I2} = -148.05$ |
| $M_{I3} = 0.2593$ | $PH_{I3} = -42.15$ |

Given Inductive Reactances

| | |
|---|---|
| $X_{L13} = 100$ | 2.27 uH |
| $X_{L12} = 5$ | 0.1136 uH |
| $X_{L23} = 20$ | 0.4547 uH |

Initial Values and Step Sizes

| | |
|---|---|
| $X_{C13} = X_{1o} = 12.0$ | $S_{1o} = 10.0$ |
| $X_{C31} = X_{2o} = 34.0$ | $S_{2o} = 10.0$ |
| $X_{C12} = X_{3o} = 50.0$ | $S_{3o} = 10.0$ |
| $X_{C21} = X_{4o} = 9.0$ | $S_{4o} = 1.0$ |
| $X_{C23} = X_{5o} = 78.0$ | $S_{5o} = 10.0$ |
| $X_{C32} = X_{6o} = 134.0$ | $S_{6o} = 10.0$ |

Hillclimber Parameters

$I = 600$ $S_{max} = 10000.0$ $S_{min} = 0.000001$

Initial Value of Hill Height

$HH_1 = 2.294484$

Calculated Currrent Distribution

$M_{b1n} = 1.0000$ $PH_{b1n} = 0.000$

$M_{b2n} = 0.9172$ $PH_{b2n} = -143.9$

$M_{b3n} = 0.1226$ $PH_{b3n} = -58.9$

Resulting Capacitive Reactances and Capacitance at 7 MHz

| | | |
|---|---|---|
| $X_{C13}$ = | 154022 | 0.147 pF |
| $X_{C31}$ = | 0.2181 | 104 nF |
| $X_{C12}$ = | 9.59E-2 | 237 nF |
| $X_{C21}$ = | 1.49E-2 | 1.52 uF |
| $X_{C23}$ = | 4.5428 | 5.00 nF |
| $X_{C32}$ = | 0.1136 | 200 nF |

The final designed problem to be presented uses the hillclimber to adjust all nine variables of the PI networks until the best approximation of the desired radiation pattern is realized. The resulting radiation pattern is shown in Figure 34.

Desired Current Distribution in Magnitude and Phase

| | |
|---|---|
| $M_{I1} = 1.000$ | $PH_{I1} = 0.00$ |
| $M_{I2} = 0.8480$ | $PH_{I2} = -148.05$ |
| $M_{I3} = 0.2593$ | $PH_{I3} = -42.15$ |

There are no Given Reactances

Initial Values and Step Sizes

| | |
|---|---|
| $X_{C13} = X_{1o} = 23.0$ | $S_{1o} = 10.0$ |
| $X_{C31} = X_{2o} = 34.0$ | $S_{2o} = 10.0$ |
| $X_{C12} = X_{3o} = 13.0$ | $S_{3o} = 10.0$ |
| $X_{C21} = X_{4o} = 45.0$ | $S_{4o} = 10.0$ |
| $X_{C23} = X_{5o} = 56.0$ | $S_{5o} = 10.0$ |
| $X_{C32} = X_{6o} = 78.0$ | $S_{6o} = 10.0$ |
| $X_{L13} = X_{7o} = 7.0$ | $S_{7o} = 1.0$ |
| $X_{L12} = X_{8o} = 132.0$ | $S_{8o} = 10.0$ |
| $X_{L23} = X_{9o} = 58.0$ | $S_{9o} = 1.0$ |

Hillclimber Parameters

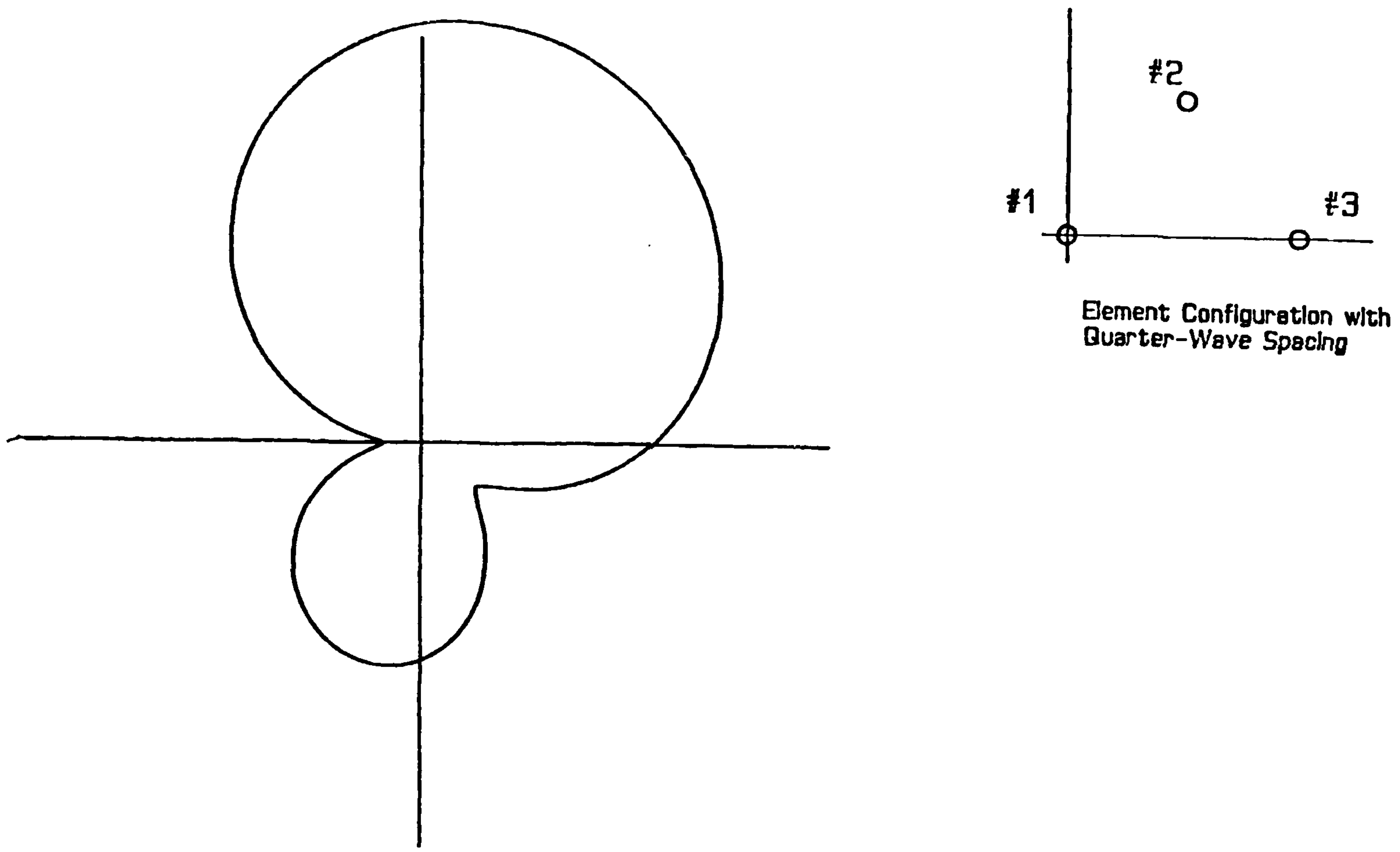


FIGURE 34 - Pattern from 9 Parameter Adjustment

$I = 900$   $S_{max} = 10000.0$   $S_{min} = 0.000001$

Initial Value of Hill Height

$HH_1 = 2.295662$

Calculated Currrent Distribution

$M_{b1n} = 1.0000$   $PH_{b1n} = 0.000$

$M_{b2n} = 0.8582$   $PH_{b2n} = -148.6$

$M_{b3n} = 0.2918$   $PH_{b3n} = -4.127$

Resulting Capacitive Reactances and Capacitance at 7 MHz

| | | |
|---|---|---|
| $X_{C13}$ | = -2.36E-2 | 536 pH |
| $X_{C31}$ | = 0.18658 | 121 pH |
| $X_{C12}$ | = -2.93E-3 | 666 pH |
| $X_{C21}$ | = 0.54461 | 41.7 nF |
| $X_{C23}$ | = 2.47199 | 9.19 nF |
| $X_{C32}$ | = 611.242 | 37.2 pF |
| $X_{L13}$ | = -32.495 | 699 pF |
| $X_{L12}$ | = 6.09429 | 138 nH |
| $X_{L23}$ | = 698.966 | 15.9 uH |

Upon comparing the desired radiations pattern (Figure 31) with the calculated radiation patterns (Figures 32,33, and 34) one can see that for this particular set of current distributions a fairly good pattern was generated from all three types of problems. The adjustment of the three inductive reactances resulted in a radiation pattern with a null direction that is about 5 degrees away form the desired direction. The main lobe direction is quite close to the desired direction and the large beamwidth of the main lobe is quite evident.

The problem which holds the inductive reactances constant while adjusting the capacitative reactances performed better than the first problem. The null direction was within 0.5 degrees of the desired directions and the main lobe was clearly in the 60 degree direction. The final problem's (adjustment of all nine PI network parameters) results were nearly as good as the second problems results (Figure 33) except that the null depth was not as deep (see Figure 34).

## C. Conclusions

The design problems above were all examined using other input current distributions corresponding to other radiation patterns. The results were generally very good. Excellent approximation to the desired pattern was obsevered in about 50% of the patterns desired, rough approximation was realized in about 30%, and about 20% of the patterns desired were not able to be realized at all. The complexity of these designed problems, the use of the unique optimization routine of Chapter Six, and the small real components resulting from the design method of Chapter Five leads one to believe that finding the desired current distribution using purely reactive interconnection networks is indeed a difficult problem.

# VIII. CONCLUSIONS AND RECOMMENDATIONS

The beam steering procedure presented in Chapter Two worked quite well on all the radiation patterns the author desired. The program MAX_RAD listed in Appendix A was run inside two Do Loops with one loop incrementing the $\Theta_m$ direction and the other incrementing the $\theta_n$ direction. Since the null direction is realized by solving a set of linear equations for the current amplitudes it will always be in the direction desired. However, additional nulls can be produced in unspecified directions. n the other hand, the current phases provide a means of controlling the direction of maximum radiation.

The hillclimber of Chapter Six performed a maximization procedure on the current phases of the three elements in the antenna system of Figure 5. Since there was just three elements in the array the potential degree of maximization was minimal because the null is already defined and the remaining radiation always had a very broad beamwidth and was usually already fairly close to the desired direction.

This beam steering procedure can be modified in order to handle as many elements as desired. This procedure provided the author with the current distributions needed for designing the interconnection medium of this thesis.

The analysis of the phased array antenna system proposed in this thesis was demonstrated in Chapter Four. The reader was provided with the electrical and physical characteristic to be used by the author in his investigations. As was shown in the results, after assuming some reasonable constant values for the PI network parameter and solving the linear complex equations describing the system, the resulting current distributions on the array elements are of no use to a potential user (see Figure 24). However, the idea of adjusting the PI network parameters until a useful current distribution is realized makes the analysis equations of Chapter Four the basis for the design technique of Chapter Seven.

In the design of the phased array antenna system of Figure 5 one assumes the desired current distributions is known through some form of beam steering technique as demonstrated in Chapter Two. Therefore, the values of the PI network parameters are the unknown variables desired by the designer of the system. Chapter Five discusses the nonlinearity of the systems' equations in this situation and demonstrated a linearization procedure and technique for solving these linearized complex equations (see equations 5.27-5.40). The resulting PI network parameters from this technique are not very practical. The designed parameters contain very small real components and the reactive components are also quite small. The capacitors that would correspond to the ohm values of these reactances are very large at 7-Mhz.

Another draw back to the design technique is the fact that one must chose seven of the nine PI network parameters. This situation immensely limits this techniques fexibility. Another problem with this technique is that the seven PI network parameters that must be specified by the user can be no greater in magnitude than 10 ohms reactive. Any time this technique is tried with parameters larger than 10 ohms in value, it diverges ( no solution is obtained).

However, of all the radiation patterns this technique was run with, using parameters of 10 ohms or less, a 90% convergence rate was achieved in less than 5 iterations of the iterative matrix solver subroutine DESIGN listed in Appendix C.

Chapter Six was completely devoted to the development and demonstration of a direct search optimization routine developed by the author for application in a design technique discussed in Chapter Seven. The hillclimber subroutine HCSUBR in Appendix D was developed to the point where it now functions as a n-variable optimization routine requiring user input of the initial guess and step size for each variable along with the range of the variables step sizes and the number of iterations desired. Also a user must create a subroutine containing the functions, inequalities, or system to be optimized. An example of this subroutine's form is shown in Appendix D under the name FUNC.

It is the author's opinion that this hillclimber performs very well, and with some refinement and continued study, could become even more competitive with some of the more traditional direct search optimization routines ( Simplex and Rosenbrock). This routine has been used on quadratic and transcendental functions of several variables as well as in the beam steering procedure of Chapter Two and the design technique of Chapter Seven. From observation of the hillclimber's performance in the design technique of Chapter Seven and on some of the more complex nonlinear quardratic functions, the routine seems to be dependent on a good initial guess for the magnitude of the variables in order to reach an acceptable solution.

Chapter Seven developed and demonstrated a promising technique for the design of interconnection networks for the three element phased array antenna system around which this thesis has evolved. This technique uses the 15 complex linear equations of Chapter Four as the system of equations that when solved result in the current distributions on the three elements for a particular set of PI network parameters. These current distributions will be optimized to some desired magnitude and phase using the hillclimber of Chapter Six. Basically the user inputs an initial guess for the PI network parameters and the hillclimber adjusts these parameters until the desired current distributions result from the solution of the 15 analysis equations.

The first form of this design technique to be investigated was the six variable optimization of the six capacitive reactances of the PI network parameters. The idea behind this problem was the practical considerations of using adjustable capacitors in conjunction with three constant inductors if this proposed array was ever constucted. The results from the investigation of this problem were mixed. Given enough iteration, a reasonable approximation of the desired pattern was achieved for about 50% of the radiation patterns this procedure was tried on. The author observed that if the $\Theta_m$ and $\Theta_n$ directions were within 45 degrees of one another, the hillclimber had problems trying to produce a reasonable pattern. Also the hillclimber tended to adjust the capacitative reactances to negative values which indicated the system wanted an inductor at that point instead of a capacitor. The author then put constraints on the range of the variables to try and prevent this occurrence. But, this caused problems for some examples in the number of iterations needed for convergence. In other problems these magnitude constraints even prevented some examples that were converging before the constraints from producing usable patterns at all.

The design technique was tried on several combinations of PI network parameters, besides the three problems discussed in Chapter Seven, and the results were quite similar to the results already shown. The design problem which uses all nine PI network Parameters seems to be of most promise from a point of flexiblitiy in the systems choice of network configuration. Even though more iterations of the hillclimber are needed because of the number of variables in the problem, the added flexibility outweighs this consideration.

The author suggests further investigation of the design of interconnection networks for phased array antenna system in the following areas: formulating analysis equations for larger arrays to see how lossless reactive networks function as the interconnection medium on a larger scale; investigation of a more suitable interconnection medium, possibly some form of reactive network where the parameters can function as both inductor and capacitor; and the method of feeding this unqiue array must be explored in further detail because how the system is powered is a very important consideration.

## REFERENCES

1. Oliver, A. A. and Knittel, G. H., "Phased Array Antennas," Proceedings of the 1970 Phased Array Antenna Symposium (New York), 1972, p. 9.

2. Stuckman, B. E., Pattern Optimization of Phased Array Antenna Systems, Oakland University School of Engineering Graduate Project, Oakland, Mi., 1984, pp. 2-9.

3. Stuckman, B. E. and Hill, J. C., " Pattern Steering in HF Phased Array Antennas," Proceedings of the 29th Midwest Symposium on Circuits Systems, Lincoln, Nb., 1986, p. 4.

4. Maron, M. J., Numerical Analysis: A Practical Approach, Macmillan Publishing Company, Inc., New York, 1982, pp. 48-51.

5. Hill, J. C., Interim Report on Flexible Vehicle Parameter Identification, June 1970, General Motors Corporation, Chevrolet Motor Divison, Vehicle Dynamics Group, Oakland, Mi.

6. Balanis, C. A., Antenna Theory: Analysis and Design, Harper and Row, New York, 1982, pp. 204-205

7. Noll, M., 73 Vertical, Beam and Triangle Antennas, Editors and Engineers, LTD., New Augusta, In., 1975, p. 38.

8. Ibid.

9. Ibid. p. 39.

10. Stuckman and Hill, p. 1.

11. Ibid. p.2.

12. Ibid.

13. Ibid.

14. Ibid.

15. Laport, A., Radio Antenna Engineering, Mcgraw-Hill New York, 1952, p. 407.

16. Terman, F. E., _Radio Engineers' Handbook_, McGraw-Hill, New York, 1943, p. 212.

17. _Ibid_.

18. Stuckman, pp. 11-12.

19. Terman, p. 782.

20. Stuckman, p. 6.

21. _Ibid_.

22. _Ibid_., p. 13.

23. Terman, p. 172.

24. Terman, p. 185.

25. Terman, p. 777.

26. Maron, p. 48.

27. Thomas, G. B. and Finney, R. L., _Calculus and Analytic Geometry_, 5th ed., Addison-Wesley, Massachusetts, 1979, p. 789.

## BIBLIOGRAPHY

1. Balanis, Constantine A., Antenna Theory Analysis and Design, New York, Harper and Row, 1982.

2. Cunningham, John E., The Complete Broadcast Antenna Handbook - Design, Installation, Operation and Maintenance, Tab Books, PA., 1977.

3. Headquarters Staff of the American Radio Relay League, The Radio Amateur's Handbook, The American Radio Relay League, Inc., Newlington, CN., 1973.

4. Hill, J. C., Interim Report on Flexible Vehicle Parameter Identification, June 1970, General Motors Corporation, Chevrolet Motor Divison, Vehicle Dynamics Group, Oakland, Mi.

5. Hill, J. C. and Downs II, J. D., "Interconnection Networks for Closely Coupled HF Phased Array Atennas," Proceedings of the 1986 Midwest Symposium on Circuits and Systems, Lincoln, NB.

6. Hudson, J. I., Adaptive Array Principles, Peter Peregrinus Ltd., New York, 1981.

7. King, Ronald W. P., Richard B. Mack, and Sheldon S. Sandler, Arrays of Cylindrical Dipoles, University Press, New York, 1968.

8. Kuecken, John A., Antennas and Transmission Lines, Howard W. Sams and Co., Indianapolis, IN., 1969.

9. Laport, Edmund A., Radio Antenna Engineering, McGraw-Hill, New York, 1952.

10. Ma, M. T., Theory and Application of Antenna Arrays, John Wiley and Sons, New York, 1974.

11. Noll, Edward M., 73 Vertical, Beam, and Triangle

Antennas, Editors and Engineers, Ltd., New Augusta, IN., 1975.

12. Oliner, Dr. Arthur A., and Dr. George H. Knittel, eds., Phased Array Antennas, Artech House, Inc., Dedam, Ma., 1972.

13. Terman, Frederick Emmons, Radio Engineers' Handbook, McGraw-Hill, New York, 1943.

14. Thomas, G. B. and Finney, R. L., Calculus and Analytic Geometry, 5th ed., Addison-Wesley, Massachusetts, 1979.

15. Stuckman, B. E., Pattern Optimization of Phased Array Antenna Systems, Oakland University School of Engineering Graduate Project, Oakland, Mi., 1984.

16. Stuckman, B. E. and Hill, J. C., "Pattern Steering in HF Phased Array Antennas," Proceedings of the 1986 Midwest Symposium on Circuits and Systems, Lincoln, NB., 1987.

# APPENDIX A

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       Program Array Pattern Plotter (HilPAT)
c
c       Version 2.1
c
c       Written by: Joseph D.Downs II                      Date:May 1986
c
c
c               This program allows a user to input (Nant) antenna elements
c       at any locaion in the XY plane.  The spacing between each antenna
c       is also a variable that can be input by the user.  The user is
c       prompted for the excitation amplitude and phase for each element
c       and the resulting radiain pattern is plotted using a call to the
c       module PPLOT.FOR.
c

        dimension amp(50),ph(50),x(50),y(50),xy(360,2),er(50),ei(50),Etot(360)
        real k,d,p,length,Emax,db
        integer nant

c       open(unit=7,file='hilpat',status='new')

        Emax=0.0
        pi=2.0*asin(1.0)
        k=2.0*pi

        write(5,*)' Input length factor between elements (length)'
        read(6,*)length

        write(5,*)'Input # of Elements'
        read(6,*)nant

        do J=1,nant
                write(5,*)'Input position of each element #',J
                read(6,*)x(J),y(J)
        end do

        do J=1,nant
                write(5,*)'Input amplitude & Phase (deg.) of element #',J
                read(6,*)amp(J),Ph(J)
                Ph(J)=Ph(J)*pi/180.0
        end do

        do I=1,360

                Theta=float(I)*pi/180.0

                do J=1,nant

                        if(X(J).eq.0.0)then
                                fphi=sign(pi/2.0,Y(J))
                                goto 50
                        endif

                        fphi=atan(Y(J)/X(J))
                        if(X(J).lt.0.0)fphi=fphi+pi
 50                     D=sqrt(X(J)**2.+Y(J)**2.)*cos(fphi-Theta)
                        P=(D/length)*K
```

```
      write(7,*)'bx1=',x(1),' bx2=',x(2),' bx3=',x(3)
      write(5,*)'amp1=',amp(1),' amp2=',amp(2),' amp3=',amp(3)
      write(5,*)'bx1=',x(1),' bx2=',x(2),' bx3=',x(3)


      ph(1)=x(1)*pi/180.0
      ph(2)=x(2)*pi/180.0
      ph(3)=x(3)*pi/180.0


      write(7,*)'END --Convergence'
      write(7,*)'oerr=',oerr,' ph1=',ph(1),' ph2=',ph(2),' ph3=',ph(3)
      write(7,*)'                '
      write(7,*)' The Real & Imag currents equal'

      do I=1,3
              cur123(I)=amp(I)*cos(ph(I))+cj*amp(I)*sin(ph(I))
              write(7,*)' The curr #,',I,' equals',cur123(I)
      end do

      write(7,*)'                '
c
c     call patt3(amp,ph,kl,pi,length)

      CLOSE(7)
      STOP
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C       PROGRAM MAX_RAD
C
C               THIS MAIN PROGRAM CALLS THE HILCLIMBER SUBROUTINE TO
C       TO ADJUST THE ELECTRICAL PHASE OF THE ELEMENTS OF THE THREE
C       ELEMENT ARRAY IN FIGURE 1 UNTIL THE RADIATION ON THE DIRECTION
C       Om IS MAXIMIZED.  THE HILLCLIMBER SUBROUTINE MAX_HC CALLS THE
C       THE SUBROUTINE FUNC_MAX WHICH IN TURN CALLS THE NULL AND FFMAG
C       ROUTINES.  THE NULL MODULE CALCULATES THE AMPLITUDES OF THE
C       CURRENT CORRESPONDING TO THE RADIATION PATTERN WITH THE DESIRED
C       NULL DIRECTION On.
C
C
C

        REAL x(3),dincrx(3),pi,THmax,THnull,acur,bamp1,bamp2,bamp3
        REAL amp(3),ph(3)

        complex cur123(3),cj

        integer time,IHOW

        open(unit=7,file='max_rad',status='new')

        write(5,*)'How long will iterations for this run (Int)'
        read(6,*)time
c
c
        cj=(0.0,1.0)
        pi=2.0*asin(1.0)

        write(5,*)'Please input guesses for Phase and step'

        do ir=1,3

                read(6,*)x(ir),dincrx(ir)
                write(7,*)'Initial Guess and Step',x(ir),dincrx(ir)

        end do

        write(5,*)'Input main lobe direction(THmax)'
        read(6,*)THmax
        write(7,*)'The main lobe direction(THmax) is ',Thmax
        THmax=THmax*pi/180.0
        write(5,*)'Input null direction (THnull)'
        read(6,*)THnull
        write(7,*)'The null direction (THnull) is',Thnull
        THnull=THnull*pi/180.0

        write(5,*)' when do you want output of iterations to begin?'
        read(6,*)IHOW
c
c
c

        CALL MAX_HC(THNULL,THMAX,X,AMP,TIME,DINCRX,IHOW)

        write(7,*)'amp1=',amp(1),' amp2=',amp(2),' amp3=',amp(3)
```

```
                er(J)=amp(J)*cos(ph(J)+P)
                ei(J)=amp(J)*sin(ph(J)+P)

        end do

        do J=2,nant
                er(1)=er(1)+er(J)
                ei(1)=ei(1)+ei(J)
        end do


Etot(I)=Sqrt(er(1)**2.+ei(1)**2.)
if(Etot(I).gt.Emax)Emax=Etot(I)

end do

write(5,*)' Emax=',Emax


do I=1,360

        Theta=float(I)*pi/180.0
        xy(I,1)=ETOT(I)*COS(theta)
        xy(I,2)=ETOT(I)*SIN(theta)

end do

call plot(xy)

stop
end
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SUBROUTINE MAX_HC
C
C               THIS SUBROUTINE IS THE N-VARIABLE VERSION OF THE
C       THE HILL CLIMBER OPTIMIZATION ROUTINE DEVELOPED BY THE THESIS
C       STUDENT.  THIS IS THE MOST GENERAL PURPOSE ROUTINE AND IT
C       IS USED FOR TESTING THE OPTIMIZATION PROCESS ON GIVEN
C       QUADRATIC, LOGRYTHMIC, AND TRANSCENDENTAL
C       FUNCTIONS.  IT IS USED FOR MAXIMIZATION INSTEAD OF MINIMIZATION.
C
C
C
C

        SUBROUTINE MAX_HC(THnull,THMAX,x,AMP,time,dincrx,ihow)


        REAL OERR,ERR,what,acur,acur10
        real mlimit,mlimit10,Merr(40000),Ediff
        real arrx(10,40000),x(10),dx(10),dincrx(10)
        real xsml(10),xlrg(10)
        real bestx(10),THnull,THMAX,AMP


        integer ch,wait,wait2,nvar,nvar2,ixdir(10),idch(10)
        integer te,td,time,mdist,ihow

C
C       INITIALIZATION OF COUNTERS
C
        ACUR=0.001
        MLIMIT=1000.0
        NVAR=3
        MDIST=10*NVAR

        ch=1
        oerr=1.0E-10
        I=0
        K=0
        J=0
        L=0
        M=0

C
C       VARIABLES USED IN STEP SIZE RESTRAINT CODE
C

        mlimit10=mlimit*10.0
        acur10=acur/10.0

C
C       VARIABLES OF THE ROUTINE THAT ARE BASED ON THE NUMBER
C       OF VARIABLES BEING ADJUSTED BY THE PROCESS.
C


        nvar2=nvar*2
        wait=nvar*4
        wait2=nvar*2+1
```

```
c
c       INITIALIZATION OF A COUNTERS AND DIRECTION INDICATORS THAT DESCRIBE
C       THE PARTICULAR STATE OF EACH VARIABLE DURING THE OPTIMIZATION.
c

        do io=1,nvar

                dx(io)=1.0
                ixdir(io)=1
                xsml(io)=0.0
                xlrg(io)=0.0
                idch(io)=0

        end do

c
c       This begins the Hclimber by calling fuction to be optimized
c

 100    Call Func_MAX(THNULL,THMAX,err,x,nvar,AMP)

c
c       This is iteration counter used in Hclimber
c

        I=I+1

C
C       THE ARRAY MERR(I) CONTAINS ALL THE VALUES OF THE HILL HEIGHT.
C       EDIFF IS THE DIFFERENCE BETWEEN THE PRESENT HILL HEIGHT AND THE
C       PREVIOUS HILL HEIGHT AND OEDIFF IS THE DIFFERENCE BETWEEN THE
C       PRESENT HILL HEIGHT AND THE HILL HEIGHT OF LOWEST MAGNITUDE FOUND
C       TO DATE.
C

        Merr(I)=err
        Ediff=abs(Merr(I)-Merr(I-1))
        OEdiff=abs(err-oerr)

c
c       Printing option that prints out error,I, and Network prameters.
c

        if(I.ge.Ihow)then
        write(7,*)' '
        WRITE(7,15)I,err,ch,ediff
 15     Format(/' I= ',I5,3x,'Error= ',E16.6,3x,'ch=',I2,2x,'Ediff=',E16.6)
        write(7,*)' '
        write(7,17)x(1),x(2),x(3)
 17     Format(' X1= ',F16.6,3x,'x2=',F16.6,3x,'x3= ',F16.6)
        write(7,*)'  '

        write(7,*)' Step sizes ...........'
        write(7,*)' '

c       write(7,16)dincrx(1),dincrx(2),dincrx(3)
```

```
c  16   Format('i1=',F10.4,2x,'i2=',F10.7,2x,'i3=',F10.7)

        write(7,*)'  '
        write(7,*)' Direction ...........'
        write(7,*)'  '

        write(7,19)dx(1),dx(2),dx(3)
   19   Format('d1=',F4.1,1x,'d2=',F4.1,1x,'d3=',F4.1)

        write(7,*)'  '

        endif

c
c       This loop loads the storage array with all values of variables through
c       out the iteration sequence.
c

        do ix=1,nvar

               arrx(ix,I)=x(ix)

        end do

c
c       These two do loops determine determine when a variable needs to be
c       reduced in size when it has gotten relatively close to an answer
c       and begins jumping around it.
c

        IF(I.GE.wait)then

               do ia=1,nvar

C
C                     J,K,L,M AND MM ARE USED TO COUNT THE LENGTH OF
C                     A PERIODIC PATTERN.
C

                      j=l+1
                      k=l + wait2
                      m=0
                      mm=0

               do ib=1,nvar2

C
C                     THIS STATEMENT DETECTS THE PERIODIC PATTERN.
C

                      if(arrx(ia,j).eq.arrx(ia,k))m=m+1
                      j=j+1
                      k=k+1

               end do

C
C              THIS STATEMENT PREVENTS PERIODIC PATTERN FROM BEING
C              DETECTED ONE RIGHT AFTER ANOTHER.
C
```

```
                if(l.le.idch(ia))goto 1
c
C        IF PATTERN IS DETECTED THEN THE STEP SIZE OF THE PERIODIC VARIABLE
C        IS REDUCED IN MAGNITUDE BY A FACTOR TEN AND THE MAGNITUDE OF
C        THE VARIABLE IS RESET TO THE BEST VALUE FOUND SO FAR IN THE PROCESS.
C
C

                if(m.eq.nvar2)then
                        idch(ia)=1 + wait2
                        x(ia)=bestx(ia)
                        dincrx(ia)=dincrx(ia)/10.0

c                       write(7,*)'***********--variable is',ia
c                       write(7,*)'Decreased increment of X,dincrx=',dincrx(ia)
c                       write(7,*)'X= ',X(ia)

                endif

   1    end do          .

        l=l+1

        endif

c
c       This loop keeps the step size of the variables from going out of limit.
c

        do if=1,nvar

        if(dincrx(if).le.acurl0)then
                dincrx(if)=acur

c               write(7,*)'----------------------------------------'
c               write(7,*)'dincrxl has gone to low, increased it to',dincrxl

        endif

        end do

c
c       This block of code determines when to increases the size (mag.) of the
c       variable based on how far it has climber in one direction.
c

        do ic=1,nvar

C
C               THIS STATEMENT CHECKS THE VARIABLES DIRECTION INDICATOR
C               AND COUNTS HOW MANY ITERTION THEY HAVE MOVED IN THE SAME
C               DIRECTION.
C
                if(ixdir(ic).eq.1)then
                        xlrg(ic)=xlrg(ic)+1
                else
                        xlrg(ic)=0
                endif
```

```
C
C               THIS STATEMENT DOES THE SAME AS THE ABOVE FOR THE
C               OPPOSITE DIRECTION.
C

                if(ixdir(ic).eq.0)then
                        xsml(ic)=xsml(ic)+1
                else
                        xsml(ic)=0
                endif

C
C               THE NEXT TWO IF-THEN'S INCREASE THE STEP SIZE OF THE
C               VARIABLES IF THEY HAVW MOVED CONTINUOUSLY IN THE
C               SAME DIRECTION.
C

                if(xlrg(ic).eq.mdist)then
                        dincrx(ic)=dincrx(ic)*10.0

c                       Write(7,*)'############--variable is',ic
c                       Write(7,*)'Increased increment of X,dincrx=',dincrx(ic)

                        xlrg(ic)=0
                endif
c
c
                if(xsml(ic).eq.mdist)then
                        dincrx(ic)=dincrx(ic)*10.0

c                       write(7,*)'############ --variable is',ic
c                       write(7,*)'Increaesd increment of X,dincrx=',dincrx(ic)

                        xsml(ic)=0
                endif
        enddo

C
C       IF THE STEP SIZE HAS GONE ABOVE THE LIMIT SPECIFIED BY THE USER
C       THIS LOOP WILL REDUCE THE STEP SIZE BY A FACTOR OF TEN.
C

        do if=1,nvar

        if(dincrx(if).ge.mlimit10)then

                dincrx(if)=mlimit
c               write(7,*)'STEP SIZE GONE ABOVE LIMIT'
c               write(7,*)'  '
        endif

        end do

c
c       This block takes the first initial step in a positive incrmental
c       direction for X(1).
c

        IF(I.EQ.1)THEN
```

```
                x(1)=x(1) + dx(1)*dincrx(1)
                write(7,*)'  '
                WRITE(7,*)'Initial HH =',err
                if(err.lt.oerr)then
                        oerr=err
                endif
                GOTO 100
        ENDIF

c
c       This block of code changes the variable back to its previous value
c       if changing it caused the hill height to increase.  This block also
c       adjusts the variables based on the direction indicators and step size
c       magnitudes by evaluating the magnitude of the hill height.
c


        do id=1,nvar

C
C               THIS OUTER IF-THEN STATEMENT PERFORMS THE VARIABLE ADJUSTMENTS
C               WHEN THE ROUTINE IS OPERATING ON THE LAST VARIABLE IN THE
C               FUNCTION.
C
                if(ch.eq.nvar)then

C
C               THIS IF-THEN REDUCES THE LAST VARIABLE BACK TO IT'S
C               PREVIOUS VALUE IF THE PRESENT HILL HEIGHT EQUALS THE
C               PREVIOUS HILL HEIGHT.  IT ALSO IMMEDIATELY ADJUSTS
C               THE FIRST VARIABLE TO ITS NEW POSITION.  THE STEP
C               SIZE OF THE LAST VARIABLE IS INCREASED BY A FACTOR
C               OF TEN.
C

                        IF(EDIFF.eq.0.0)THEN

                                x(nvar)=x(nvar)-dx(nvar)*dincrx(nvar)
                                x(1)=x(1)+dx(1)*dincrx(1)

                                DINCRX(NVAR)=DINCRX(NVAR)*10.0

c                       WRITE(7,*)'ERR  - PREVIOUS ERR LT 1E-6,Bx=',x(nvar)
c                       WRITE(7,*)'I=',I,' id=',id,' step=',dincrx(nvar)

                        goto 70
                        ENDIF

C
C                       THE SAME AS ABOVE IS PERFORMED WHEN THE PRESENT
C                       HILL HEIGHT EQUALS THE BEST HILL HEIGHT FOUND
C                       TO DATE.
C

                        IF(OEDIFF.eq.0.0)THEN

                                x(nvar)=x(nvar)-dx(nvar)*dincrx(nvar)
                                x(1)=x(1)+dx(1)*dincrx(1)
                                DINCRX(NVAR)=DINCRX(NVAR)*10.0
```

```
c                         WRITE(7,*)'Oerr and Err are = to e-6'
c                         write(7,*)'EEEERRRRRRRRRRR = OOOOeeerrrr'
c                         WRITE(7,*)'I=',I,' id=',id,' step=',dincrx(nvar)

                          goto 70

                  ENDIF

C
C                 IF THE HILL HEIGHT IS LOWER THEN THE LAST VARIABLE
C                 IS KEPT AT IT'S NEW MAGNITUDE AND THE FIST VARIABLE
C                 IS ADJUSTED.
C

                  if(err.gt.oerr)x(1)=x(1)+dx(1)*dincrx(1)

C
C                 IF THE HILL HEIGHT IS GREATER THE LAST VARIABLE
C                 IS REDUCED AND FIRST VARIABLE IS ADJUSTED IMMEDIATELY.
C

                  if(err.lt.oerr)then
                         x(nvar)=x(nvar)-dx(nvar)*dincrx(nvar)
                         x(1)=x(1)+dx(1)*dincrx(1)
                  endif
          goto 70
          endif

C
C         THE FOLLOWING IF-THEN STATEMENTS ARE FOR ALL OF THE ITERATIONS
C         BESIDES ADJUSTMENT OF THE LAST VARIABLE ( IN OTHER WORDS
C         MOVEMENT OF THE VARIABLES IN SEQUENCE 1,2,3,..... N, BUT
C         NOT N.  MOVEMENT OF THE LAST VARIABLE (N) IS HANDLED ABOVE.
C         CH TRACKS WHICH VARIABLE IS CURRENTLY BEING ADJUSTED.
C

          IF(CH.EQ.ID .AND. EDIFF.eq.0.0)THEN

                 TE=ID+1
                 x(id)=x(id)-dx(id)*dincrx(id)
                 x(te)=x(te)+dx(te)*dincrx(te)
                 DINCRX(ID)=DINCRX(ID)*10.0

c                WRITE(7,*)'ERR  - PREVIOUS ERR LT 1E-6,Bx=',x(id)
c                WRITE(7,*)'I=',I,' id=',id,' step=',dincrx(nvar)

          goto 70

          ENDIF


          IF(CH.EQ.ID .AND. OEDIFF.eq.0.0)THEN

                 TE=ID+1
                 x(id)=x(id)-dx(id)*dincrx(id)
                 x(te)=x(te)+dx(te)*dincrx(te)
                 DINCRX(ID)=DINCRX(ID)*10.0

c                WRITE(7,*)'Oerr and Err are = to e-6'
c                write(7,*)'EEEERRRRRRRRRRR = OOOOeeerrrr'
```

```
c                    WRITE(7,*)'I=',I,' id=',id,' step=',dincrx(nvar)

                     goto 70
              ENDIF


              IF(ch.eq.id .and. err.gt.oerr)then

                     td=id+1
c                    if(id.eq.nvar)td=1
                     x(td)=x(td)+dx(td)*dincrx(td)

              endif


              If(ch.eq.id .and. err.lt.oerr)then
                     td=id+1
                     x(id)=x(id)-dx(id)*dincrx(id)
                     x(td)=x(td)+dx(td)*dincrx(td)

              endif

       end do

c
c      This loop changes the variables direction based on the magnitude of
c      the hill height.  If the variables adjustment causes the hill height
c      not to be reduced below the current best hill height then the
c      particular variables direction indicators are reversed so the variable
c      can move in the other direction.
c


  70   do ie=1,nvar

              if(ch.eq.ie .and. ixdir(ie).eq.1 .and. err.lt.oerr)then
                     ixdir(ie)=0
                     dx(ie)=-1.0
                     goto 20
              endif

              if(ch.eq.ie .and. ixdir(ie).eq.0 .and. err.lt.oerr)then
                     ixdir(ie)=1
                     dx(ie)=1.0
                     goto 20
              endif

       end do

c
c      This block keeps track of which variable is to be changed next.
c


 20    if(ch.le.nvar)then
              if(ch.eq.nvar)ch=0
              ch=ch+1
       endif
```

```
c
c       This block records value of the best variables and current values
c       as the Hclimber proceeds thru the climb.
c

        if(err.gt.oerr)then

                write(5,*)'err=',err,' at I=',I
                oerr=err

                do ip=1,nvar
                        bestx(ip)=arrx(ip,I)
                end do

        endif

c
c       This tells machine to do another iteration
c

        IF(I.LT.time)GOTO 100

        write(7,*)'  '
        write(7,*)'HH (hill height) at end of run =',oerr

        DO IH=1,3
                X(IH)=BESTX(IH)
        ENDDO


        return
        end
```

```
SUBROUTINE FUNC_MAX(THNULL,THMAX,HILLH,X,NVAR,AMP)

REAL x(3),pi,THmax,HillH,THnull,Fmax,acur,amp(3),ph(3)

integer nvar

complex cj,cur123(3)


cj=(0.0,1.0)
pi=2.0*asin(1.0)
k=2.0*pi

ph(1)=x(1)*pi/180.0
ph(2)=x(2)*pi/180.0
ph(3)=x(3)*pi/180.0

call null(pi,ph,THnull,amp)

call FFmag(pi,amp,ph,THmax,Fmax)

HillH=Fmax

RETURN
END
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine null(pi,PH,THnull,amp)

      real x(3),y(3),ph(3),amp(3),p(3)
      real kl,THnull,c1,c2,x11,x12,pi,x21,x22,denom

      X(1)=0.0
      X(2)=0.5
      X(3)=1.0
      Y(1)=0.0
      Y(2)=0.8660254
      Y(3)=0.0

      kl= 2.0*pi/4.0

      do 20 J=1,3
              if(x(J).eq.0.0 .and. y(J).eq.0.0)goto 20
              P(J)=kl*sqrt(x(J)**2.+y(J)**2.)*cos(atan2(y(J),x(J))-THnull)
 20   continue

      P(1)=0.0

      C1=-cos(ph(1)+p(1))
      X11=cos(ph(2)+p(2))
      X12=cos(ph(3)+p(3))

      C2=-sin(ph(1)+p(1))
      X21=sin(ph(2)+p(2))
      X22=sin(ph(3)+p(3))

      denom=X11*X22-X21*X12


      if(denom.eq.0.0)then
              write(5,*)' Denom equals 0.0'

              goto 39
      endif

      amp(1)=1.0
      amp(2)=(c1*X22-X12*c2)/denom
      amp(3)=(X11*c2-C1*X21)/denom

 39   return
      end

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine FFmag(pi,amp,ph,THmax,Fmax)

      dimension x(3),y(3),ph(3),amp(3),p(3),er(3),ei(3)
      real kl,THmax,Fmax,ptotal,pi


      X(1)=0.0
      X(2)=0.5
      X(3)=1.0
      Y(1)=0.0
      Y(2)=0.8660254
      Y(3)=0.0

      kl= 2.0*pi/4.0

      do 30 J=1,3
              if(x(J).eq.0.0 .and. y(J).eq.0.0)goto 29
              P(J)=kl*sqrt(x(J)**2.+y(J)**2.)*cos(atan2(y(J),x(J))-THmax)

29    p(1)=0.0


      er(J)=amp(J)*cos(ph(J)+p(J))
      ei(J)=amp(J)*sin(ph(J)+p(J))

30    continue


      do J=2,3

              er(1)=er(1)+er(J)
              ei(1)=ei(1)+ei(J)

      end do


      Fmax=sqrt(er(1)**2.0 + ei(1)**2.0)

      call Powerin(amp,ph,ptotal)

      Fmax=Fmax/ptotal

      return
      end

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine PowerIn(amp,ph,ptotal)

      dimension amp(3),ph(3)
      complex cj,I1,I2,I3,V1,V2,V3,Zs,Z14,z1,z2,z3
      real ptotal,p1,p2,p3

      cj=(0.0,1.0)
      I1=amp(1)*cos(Ph(1)) + cj*amp(1)*sin(ph(1))
      I2=amp(2)*cos(Ph(2)) + cj*amp(2)*sin(ph(2))
      I3=amp(3)*cos(Ph(3)) + cj*amp(3)*sin(ph(3))
c
c
      Z14=(20.4,-14.18)
      Zs=(36.5,21.0)
c
c
      V1=I1*Zs + (I2+I3)*Z14
      V2=I2*Zs + (I1+I3)*Z14
      V3=I3*Zs + (I1+I2)*Z14
c
c
```

```
        p1=real(conjg(I1)*v1)
        p2=real(conjg(I2)*v2)
        p3=real(conjg(I3)*v3)
c       write(7,*)'p1=',p1,' p2=',p2,' p3=',p3
c
        Ptotal=sqrt((p1+p2+p3)/36.5)
c
c       Z1=V1/I1
c       Z2=V2/I2
c       Z3=V3/I3
c       write(7,*)'Driving point impedances equal'
c       write(7,*)z1,z2,z3
c
c
        return
        end
```

## APPENDIX B

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       PROGRAM ANAL_DESG
c
c               THIS PROGRAM SOLVES THE 15 LINEAR EQUATIONS DESCRIBING
c       THE PHASED ARRAY ANTENNA SYSYTEM OF FIGURE 5.  THE VALUES FOR
c       THE PI NETWORK PARAMETERS AND THE INPUT CURRENT ARE PASSED
c       TO THE ANALYSIS SUBROUTINE CURR WHICH RETURNS TO THIS PROGRAM
c       THE CALCULATED CURRENT DISTRIBUTIONS FOR EACH ELEMENT OF THE
c       ARRAY.
c
c

        complex ckt1(3),ckt2(3),ckt3(3),B(15),It,ITT,curr1(3),Itdesign
        complex bd(14),crr1(3)

        real amp(3),ph(3)

        open(unit=7,file='anal_desg',status='new')

        pi = 2.0*asin(1.)

        CKT1(1)=(0.0,1.0)
        CKT1(2)=(0.0,50.0)
        CKT1(3)=(0.0,560.0)

        CKT2(1)=(0.0,10.0)
        CKT2(3)=(0.0,50.0)

        CKT3(1)=(0.0,100.0)
        CKT3(3)=(0.0,5.0)

        write(5,*)'Input value for XC12 and XC13, complex'
        read(6,*)ckt2(2),ckt3(2)
c       write(7,*)'Input capacitors XC12 & XC23 equal',ckt2(2),ckt3(2)
c       WRITE(7,*)'   '

        write(7,*)' PI Network Parameters'
        write(7,*)'   '

        write(7,75)aimag(ckt1(1)),aimag(ckt1(2)),aimag(ckt1(3))
  75    format(3x,' XL13 =',2X,F6.2,' XC13 =',2X,F6.2,' XC31 =',2X,F6.2)
        WRITE(7,*)'   '

        write(7,76)aimag(ckt2(1)),aimag(ckt2(2)),aimag(ckt2(3))
  76    format(3x,' XL12 =',2X,F6.2,' XC12 =',2X,F6.2,' XC21 =',2X,F6.2)
        WRITE(7,*)'   '

        write(7,77)aimag(ckt3(1)),aimag(ckt3(2)),aimag(ckt3(3))
  77    format(3x,' XL23 =',2X,F6.2,' XC23 =',2X,F6.2,' XC32 =',2X,F6.2)
        WRITE(7,*)'   '


        write(5,*)'Input the current at the input of antenna 1'
        read(6,*)It

        write(7,*)'The input current It =',It
        WRITE(7,*)'   '
```

```
      call curr(CKT1,CKT2,CKT3,It,B)

      write(5,*)b(1),b(2),b(3)

      write(7,*)' The Resulting Current Distributions (complex)'
      write(7,*)'  '
      write(7,*)'I1=',b(1)
      write(7,*)'I2=',b(2)
      write(7,*)'I3=',b(3)
      WRITE(7,*)'    '
      write(7,*)'The Corresponding Magnitudes and Phases (degrees)'
      write(7,*)'   '

      crr1(1)=b(1)
      crr1(2)=b(2)
      crr1(3)=b(3)

      call mag_ph_norm(3,crr1,amp,ph)

      do i=1,3
              ph(i)=ph(i)*180.0/pi
      end do

      write(7,80)amp(1),amp(2),amp(3)
80    format(3x,' [I1] =',2X,F7.2,' [I2] =',2X,F7.2,' [I3] =',2X,F7.2)
      WRITE(7,81)ph(1),ph(2),ph(3)
81    format(3x,' PHS1 =',2X,F7.2,' PHS2 =',2X,F7.2,' PHS3 =',2X,F7.2)

      ITT=B(1)+B(4)+B(6)

      write(5,*)'It =I1 + I12 +I13 is equal to',ITT
      WRITE(7,*)'  '

      call design(crr1,bd,Itdesign)

      write(5,*)'XC12=',bd(13),' XC23=',bd(14)
      write(5,*)'Itdesign=',Itdesign

      WRITE(7,*)'Designed Pi Network Parameters'

      write(7,*)'XC12=',bd(13),' XC23=',bd(14)
      write(7,*)'   '

      write(7,*)'It (Input Current resulting from Design)'
      write(7,*)'Itdesign=',Itdesign

      stop
      end
c
c
c
      subroutine CURR(CKT1,CKT2,CKT3,It,B)
c
c     ***************************************************************
```

```
c
c       This program is the two-port analysis of a triangular three
c       element antenna system.
c
c       ************************************************************
c
c       ------- declare all variables----------
c
        INTEGER N,IA,M,IB,IJOB,IER
        COMPLEX A(15,15),B(15),WA(255),CKT1(3),CKT2(3),CKT3(3),XL13,XC13
        COMPLEX XC31,Z11,Z12,Z13,IT,C46,C02,PII,X1,X2,X3,X4,X5,X6
        REAL wk(15)
C       -------matrix parameters-----------
c
        IA=15
        IB=15
        N=15
        M=1
C
C       ------mutual impedances-------------
C
        Z11=(36.5,21.0)
        Z13=(20.4,-14.18)
        Z12=Z13
C
C       ------constants generated from separation distances-------
c
C
C       ------change above reals into complex #'s with zero reals
C
        C46=(0.0,46.19397663)
        C02=(0.0,.018477591)
        PII=(.382683432,0.0)
c
c       ----set constant matrix b back to zero----
c
        do 5 j=1,15
               B(j)=(0.0,0.0)
  5     continue
C
C       ------DECLARE INPUT CURRENT IT--------
C
        B(1)=IT
C
C       -----INSERT GIVEN PI NETWORK VALUES OF REACTANCE-----
C


C       ------BEGIN CALCULATING THE MATRIX "A" COEFFS-------
C
        XL13=CKT1(1)
        XC13=CKT1(2)
        XC31=CKT1(3)
C
C       CALL SUBROUTINE COEFF FOR FIRST TIME
C
        CALL COEFF(C46,C02,PII,XL13,XC13,XC31,X1,X2,X3,X4,X5,X6)
C
C       ------INSERT COEFFS INTO MATRIX "A"-------
C
```

```
          A(1,1)=(1.,0.)
          A(1,4)=(1.,0.)
          A(1,6)=(1.,0.)
          A(2,2)=(1.,0.)
          A(2,5)=(1.,0.)
          A(2,8)=(1.,0.)
          A(3,3)=(1.,0.)
          A(3,7)=(1.,0.)
          A(3,9)=(1.,0.)
C
C
          A(4,1)=Z11
          A(4,2)=Z12
          A(4,3)=Z13
          A(4,10)=X1
          A(4,11)=X2
C
C
          A(5,6)=(1.,0.)
          A(5,10)=X3
          A(5,11)=X4
C
C
          A(6,1)=Z12
          A(6,2)=Z13
          A(6,3)=Z11
          A(6,10)=X2
          A(6,11)=X5
C
C
          A(7,7)=(1.,0.)
          A(7,10)=X4
          A(7,11)=X6
C
C         -------SECOND SET OF EQUATIONS------
C
          XL13=CKT2(1)
          XC13=CKT2(2)
          XC31=CKT2(3)
C
C
C
          CALL COEFF(C46,C02,PII,XL13,XC13,XC31,X1,X2,X3,X4,X5,X6)
C
C         ------LOAD COEFFS INTO MATRIX "A"-----
C
          A(8,1)=Z11
          A(8,2)=Z12
          A(8,3)=Z13
          A(8,12)=X1
          A(8,13)=X2
C
C
          A(9,4)=(1.,0.)
          A(9,12)=X3
          A(9,13)=X4
C
C
          A(10,1)=Z12
          A(10,2)=Z11
          A(10,3)=Z13
          A(10,12)=X2
          A(10,13)=X5
C
C
          A(11,5)=(1.,0.)
          A(11,12)=X4
          A(11,13)=X6
C
C         -----THIRD SET OF EQUATIONS------
C
          XL13=CKT3(1)
          XC13=CKT3(2)
          XC31=CKT3(3)
C
C
C
C
          CALL COEFF(C46,C02,PII,XL13,XC13,XC31,X1,X2,X3,X4,X5,X6)
C
          A(12,1)=Z12
          A(12,2)=Z11
          A(12,3)=Z13
          A(12,14)=X1
          A(12,15)=X2
C
C
          A(13,8)=(1.,0.)
          A(13,14)=X3
          A(13,15)=X4
C
C
          A(14,1)=Z12
          A(14,2)=Z13
          A(14,3)=Z11
          A(14,14)=X2
          A(14,15)=X5
C
C
          A(15,9)=(1.,0.)
          A(15,14)=X4
          A(15,15)=X6
C
C
C
C
C
C         ----SET FUNCTION DEFINITION------
C
          IJOB=0
C
C         -------CALL MATRIX INVERSION IMSL ROUTINE-------
C
          CALL LEQ2C(A,N,IA,B,M,IB,IJOB,WA,WK,IER)
C
C         -----OUTPUT CALCULATED VALUES-----------
C
          write(5,*)'I1 = ',b(1)
          write(5,*,'I2 = ',b(2)
          write(5,*)'I3 = ',b(3)
```

```
      RETURN
      END
C
C     ****************************************************
C      THIS SUBROUTINE CALCULATES THE MATRIX COEFFS
C     ****************************************************
C
      SUBROUTINE COEFF(C46,C02,PII,XL13,XC13,XC31,X1,X2,X3,X4,X5,X6)
C
C     ----------DECLARE COMPLEX VARIABLES--------------
C
      COMPLEX T1,T2,X1,X2,X3,X4,X5,X6,C46,C02,PII,XL13,XC13,XC31
C
C     ----------ACTUAL COMPLEX ALGEBRA EQUATIONS-------
C
      T1=(PII+(C46/XL13))
      T2=(C02+(PII/XL13))
      X1=-(T1+(C46*XC13))
      X2=(C46/XL13)
      X3=-(T2+(PII*XC13))
      X4=(PII/XL13)
      X5=-(T1+(C46*XC31))
      X6=-(T2+(PII*XC31))
C
C
      RETURN
      END
```

## APPENDIX C

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C
C       DESIGN.FOR
C
C               THIS PROGRAM USES THE DESIGN TECHNIQUE DISCUSSED IN
C       CHAPTER 5 TO DESIGN TWO OF THE PI NETWORK PARAMETERS OF THE
C       INTERCONNECTION MEDIUM IN FIGURE 5. THE USER IS PROMPTED FOR
C       THE DESIRED CURRENT DISTRIBUTIONS AND THE OTHER PI NETWORK
C       PARAMETERS ARE ASSIGNED SOME CONSTANT MADNITUDE.  THE SUBROUTINE
C       DESIGN IS CALLED AND THE RESULTING PARAMETERS ARE LOADED INTO
C       AN OUTPUT FILE.  THE DESIGN SUBROUTINE PERFORMS THE ITERATIVE
C       PROCESS ON THE DESIGN MATRIX UNTIL INSTRUCTED TO STOP.
C
C
C
C

        dimension CKT1(3),CKT2(2),CKT3(2)
        real amp(3),ph(3)
        complex I1,I2,I3,curr(3),It,bd(14),cckt1(3),cckt2(3),cckt3(3)

        open(unit=7,file='design',status='new')

        write(5,*)'Input magnitude and phase of desired current distribution'
        write(7,*)'Input magnitude and phase of desired current distribution'

        do i=1,3
                write(5,*)'Input amp and phase for element #',i
                read(6,*)amp(i),ph(i)
                write(7,*)'For element #',i,' Mag & Phase=',amp(i),ph(i)
        end do

        call cmpn(3,amp,ph,curr)


        CKT1(1)=10.0
        CKT1(2)=1.0
        CKT1(3)=5.0
        CKT2(1)=CKT1(1)
        CKT2(2)=5.0
        CKT3(1)=CKT1(1)
        CKT3(2)=7.5

c
c

        call design(CKT1,CKT2,CKT3,curr,bd)

        write(7,*)'   '
        write(7,*)' ............NOW PERFORMING ANALYSIS..........'
        write(7,*)'   '

        It=curr(1)+bd(1)+bd(3)
        write(7,*)' It=',It

        cckt1(1)=cmplx(0.0,ckt1(1))
        cckt1(2)=cmplx(0.0,ckt1(2))
        cckt1(3)=cmplx(0.0,ckt1(3))
        cckt2(1)=cmplx(0.0,ckt2(1))
```

```
          T1=aimag(bd(13))
          T2=real(bd(13))
          cckt2(2)=cmplx(t1,t2)
          cckt2(3)=(0.0,7.0E-6)
          cckt3(1)=cmplx(0.0,ckt3(1))
          cckt3(2)=cmplx(aimag(bd(14)),real(bd(14)))
          cckt3(3)=(0.0,4.0E-7)
          write(5,*)'cckt2=',cckt2(1),cckt2(2),cckt2(3)
          write(5,*)'cckt3=',cckt3(1),cckt3(2),cckt3(3)
          write(5,*)'cckt1=',cckt1(1),cckt1(2),cckt1(3)
          call curr1(cckt1,cckt2,cckt3,It)
c
c
c         CKT1(1)=CKT1(1) + plus2
c         if(CKT1(1).LE.600.0)goto 10
c         CKT1(1)=startC1
c         CKT1(2)=CKT1(2) +plus1
c         if(CKT1(2).GE.-60.0)goto 10
c         CKT1(2)=startC2
c         CKT1(3)=CKT1(3) +plus1
c         IF(CKT1(3).GE.-60.0)GOTO 10
C
C
          close(7)
          stop
          end

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C         SUBROUTINE DESIGN
C
C                 THIS ROUTINE IS AN ITERATIVE DESIGN PROCEDURE
C         WHICH USES GIVEN CURRENT DISTRIBUTIONS FOR THE ARRAY ELEMENTS
C         AS INPUT.  IT CALCULATES TWO USER DESIGNATED PI NETWORK
C         PARAMETERS THAT SOLVE THE LINEARIZED SET OF 14 DESIGN
C         EQUATIONS.
C
C         INPUTS:  CKT1,CKT2,CKT3,CURR
C         OUTPUTS: B
C
C

          subroutine design(ckt1,ckt2,ckt3,curr,bd)

          complex a(14,14),curr(3),vt(3),I1,I2,I3,b(14),y1,y2,y3,y4
          complex c1,c2,c3,c4,V2C0,V1B0,GV1B,GV2C,wa(224),bd(14)
          integer n,ia,m,ib,ijob,ier,Q
          real wk(14),XL13,XC13,XC31,Gxc12,Gxc23,x1,x2,x3,x4,x5,x6,c46,c02,pii
          dimension CKT1(3),CKT2(2),CKT3(2)

C
C         SETS DIMENSIONS OF MATRIX
C

          ia=14
          ib=14
          n=14
          m=1
          Q=0
```

```
C
C       SETS INITIAL GUESS VALUE FOR PI NETWORK PARAMETERS AND
C       VOLTAGE VARIABLES.
C

        Gxc12=1.
        GV1B=(1.,1.)
        Gxc23=1.
        GV2C=(1.,1.)

C
C       OUTPUTS GIVEN PI NETWORK PARAMETERS TO DATA FILE.
C

        write(7,*)'THE given capacitor values are XC13,XC31,XC21,XC32'
        write(7,*)CKT1(2),CKT1(3),CKT2(2),CKT3(2)
        WRITE(7,*)'The given inductor values are equal to',CKT1(1)

C
C       INITIALIZED THE MATRIX'S OF COEFFICENTS AND CONSTANTS (A & B).
C

  100   do i=1,14
                do j=1,14
                        a(i,j)=(0.,0.)
                end do
        end do

        do i=1,14
                b(i)=(0.,0.)
        end do

C
C       THE CALL TO CMMUL PERFORMS THE COMPLEX MATRIX MULTIPLICATION
C       BETWEEN THE CURRENT DISTRIBUTIONS AND THE MUTUAL IMPEDANCES VALUES
C       WHICH RESULTS IN THE VOLTAGES AT THE BASE OF EACH ARRAY ELEMENT.
C

        call cmmul(curr,vt)

C
C       LOADS PARAMETERS OF THE FIRST PI NETWORK INTO VARIABLES THAT ARE
C       USED TO REPRESENT ALL OF THE PARAMETERS IN THE CALCULATIONS OF THE
C       COEFFICENTS OF THE UNKNOWN VARIABLES. C
C

        XL13=CKT1(1)
        XC13=CKT1(2)
        XC31=CKT1(3)

C
C       CALL TO ROUTINE THAT PERFORMS THE COEFFICENT CALCULATIONS FOR THE
C       FIRST PI NETWORK.
C

        call numb(x1,x2,x3,x4,x5,x6,c46,c02,pii,XL13,XC13,XC31)

C
C       LOADIND OF MATRIX A WITH CALCULATED COEFFICENTS.
```

```
C

      a(1,2)=(1.0,0.0)
      a(1,5)=(1.,0.)
      a(2,4)=(1.,0.)
      a(2,6)=(1.,0.)
      a(3,7)=cmplx(x1,0.)
      a(3,8)=cmplx(x2,0..
      a(4,3)=(1.,0.)
      a(4,7)=cmplx(0.,x3)
      a(4,8)=cmplx(0.,x4)
      a(5,7)=cmplx(x2,0.)
      a(5,8)=cmplx(x5,0.)
      a(6,4)=(1.,0.)
      a(6,7)=cmplx(0.,x4)
      a(6,8)=cmplx(0.,x6)

C
C     LOADS PARAMETERS OF THE PI NETWORK CONTAININD ONE OF THE
C     UNKNOWN PARAMETERS INTO THE VARIABLES THAT ARE BEING
C     USED TO REPRESENT ALL OF THE PARAMETERS IN THE CALCULATIONS
C     OF THE COEFFICENTS OF SOME OF THE UNKNOWN VARIABLES.
C

      XL13=CKT2(1)
      XC13=Gxc12
      XC31=CKT2(2)
      V1B0=GV1B

C
C     CALL TO ROUTINE THAT PERFORMS THE COEFFICENT CALCULATIONS FOR THE
C     FIRST PI NETWORK.
C

      call numb(x1,x2,x3,x4,x5,x6,c46,c02,pii,XL13,XC13,XC31)

C
C     CALCULATION OF MORE COEFFICENTS
C

      y1=-(V1B0*cmplx(C46,0.0))
      y2=-(V1B0*cmplx(0.0,pii))
      c11=c46*XC13
      c1=vt(1)-(cmplx(c11,0.0)*V1B0)
      c22=pii*XC13
      c2=-(cmplx(0.0,c22)*V1B0)

C
C     LOADIND OF MATRIX A WITH CALCULATED COEFFICENTS.
C

      a(7,9)=cmplx(x1,0.)
      a(7,10)=cmplx(x2,0.)
      a(7,13)=y1
      a(8,1)=(1.,0.)
      a(8,9)=cmplx(0.,x3)
      a(8,10)=cmplx(0.,x4)
      a(8,13)=y2
      a(9,9)=cmplx(x2,0.)
      a(9,10)=cmplx(x5,0.)
      a(10,2)=(1.,0.)
      a(10,9)=cmplx(0.,x4)
      a(10,10)=cmplx(0.,x6)
C
C     LOADS PARAMETERS OF THE PI NETWORK CONTAININD ONE OF THE
C     UNKNOWN PARAMETERS INTO THE VARIABLES THAT ARE BEING
C     USED TO REPRESENT ALL OF THE PARAMETERS IN THE CALCULATIONS
C     OF THE COEFFICENTS OF SOME OF THE UNKNOWN VARIABLES.
C
      XL13=CKT3(1)
      XC13=Gxc23
      XC31=CKT3(2)
      V2C0=GV2C
C
C     CALL TO ROUTINE THAT PERFORMS THE COEFFICENT CALCULATIONS FOR THE
C     FIRST PI NETWORK.
C
      call numb(x1,x2,x3,x4,x5,x6,c46,c02,pii,XL13,XC13,XC31)
C
C     CALCULATION OF MORE COEFFICENTS
C
      y3=-(V2C0*cmplx(c46,0.0))
      y4=-(V2C0*cmplx(0.0,pii))
      c33=c46*XC13
      c3=vt(2)-(cmplx(c33,0.0)*V2C0)
      c44=pii*XC13
      c4=-(cmplx(0.0,c44)*V2C0)
C
C     LOADIND OF MATRIX A WITH CALCULATED COEFFICENTS.
C
      a(11,11)=cmplx(x1,0.)
      a(11,12)=cmplx(x2,0.)
      a(11,14)=y3
      a(12,5)=(1.,0.)
      a(12,14)=y4
      a(12,11)=cmplx(0.,x3)
      a(12,12)=cmplx(0.,x4)
      a(13,11)=cmplx(x2,0.)
      a(13,12)=cmplx(x5,0.)
      a(14,6)=(1.,0.)
      a(14,11)=cmplx(0.,x4)
      a(14,12)=cmplx(0.,x6)
C
C     CONSTANT MATRIX B FORMATION
C
      b(1)=-curr(2)
      b(2)=-curr(3)
      b(3)=vt(1)
      b(4)=(0.,0.)
      b(5)=vt(3)
      b(6)=(0.0,0.)
      b(7)=c1
      b(8)=c2
      b(9)=vt(2)
      b(10)=(0.,0.)
      b(11)=c3
      b(12)=c4
      b(13)=vt(3)
      b(14)=(0.,0.)

C
C     CALL TO LINEAR EQUATION SOLVER IN IMSL SUBROUTINE LIBRARY.
C

      ijob=0

      call leq2c(a,n,ia,b,m,ib,ijob,wa,wk,ier)

      Q=Q+1

      write(7,*)'I (number of iterations) =',Q
      write(7,*)'   '

      write(7,90)AIMAG(B(13)),REAL(B(13)),aimag(b(14)),real(b(14))
 90   FORMAT('XC12=',2X,F9.4,' +j',F9.4,'  XC12=',2X,F9.4,' +j',F9.4)

C
C     LOADING THE NEW FOUND VALUES INTO NEXT GUESS VARIABLES FOR NEXT
C     ITERATION.
C

      GV1B=b(9)
      Gxc12=real(b(13))
      GV2C=b(11)
      Gxc23=real(b(14))

      if(Q.le.10)goto 100

      write(7,*)'End of iterations'

      do ij=1,14
              bd(ij)=b(ij)
              write(7,*)'For ij =',ij,' Bd(ij)=',Bd(ij)
      end do

      return
      end


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SUBROUTINE NUMB
C
C             THIS ROUTINE CALCULATES THE COEFFICENTS OF THE UNKNOWN
C     VARIABLES BASED ON THE INPUT CONSTANTS.  THE INPUT CONSTANTS ARE
C     BASED ON THE MAGNITUDE OF THE GIVEN PI NETWORK PARAMETERS.
C
C     INPUT:  XL13,XC13,XC31
C     OUTPUT: X1,X2,X3,X4,X5,X6
```

```
c
C

      subroutine numb(x1,x2,x3,x4,x5,x6,c46,c02,pii,XL13,XC13,XC31)
      real x1,x2,x3,x4,x5,x6,pi,pii,c46,c02,rad,t1,t11,t2,t22,con1

C
C     CALCULATIONS BASED ON THE TRANSMISSION LINE LENGTH
C

      pi=2.0*asin(1.0)
      rad=3.0*pi/8.0
      con1=sin(rad)
      pii=cos(rad)
      c46=50.0*con1
      c02=con1/50.0

C
C     CALCULATIONS BASED ON THE NETWORK PARAMETERS
C

      t1=c46/XL13
      t11=pii+t1
      x1=(t11-(c46*XC13))
      x2=-t1
      t2=pii/XL13
      t22=c02-t2
      x3=-(t22+(pii*XC13))
      x4=-t2
      x5=(t11-(c46*XC31))
      x6=-(t22+(pii*XC31))

      return
      end


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SUBROUTINE CMMUL
C
C           THIS ROUTINE PERFORMS THE COMPLEX MATRIX MULTIPLICATION
C     OF A 3X3 AND A 3X1 MATRIX.  IN PARTICULAR THIS ROUTINE CALCULATES
C     THE VOLTAGE AT THE BASE OF EACH ARRAY ELEMENT.  THE INPUT IS THE
C     CURRENT FLOWING IN EACH ELEMENT.
C
C     INPUT:  CURR
C     OUTPUT: VT
C
C

      subroutine cmmul(curr,vt)

      complex vt(3),curr(3),impe(3,3),a

C
C     MUTUAL IMPEDANCE VALUES FOR ELEMENT OF THE EQUILATERAL TRIANGULAR
C     ARRAY.
C
```

```
c
        impe(1,1)=(36.5,21.0)
        impe(1,2)=(20.4,-14.18)
        impe(1,3)=(20.4,-14.18)
        impe(2,1)=(20.4,-14.18)
        impe(2,2)=(36.5,21.0)
        impe(2,3)=(20.4,-14.18)
        impe(3,1)=(20.4,-14.18)
        impe(3,2)=(20.4,-14.18)
        impe(3,3)=(36.5,21.0)

C
C       THESE TWO DO LOOPS PERFORM THE MULIPLICATION.
C

        do k=1,3
                a=(0.0,0.0)

                        do m=1,3
                           a=a+impe(k,m)*curr(m)
                        end do

                vt(k)=a
        end do

        return
        end
```

## APPENDIX D

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       PROGRAM MAINHC
c
c               THIS PROGRAM IS USED TO TEST THE HILL CLIMBER
c       OPTIMIZATION ROUTINE ON A VARIETY OF FUNCTION AS SPECIFIED
c       BY THE USER IN THE SUBROUTINE FUNC.  THIS PROGRAM CALLS THE
c       MOST GENERAL VERSION OF THE OPTIMIZATION ROUTINE AND PASSES
c       TO IT ANY VARIABLES THAT ARE INPUT BY THE USER.  THESE VARIABLES
c       ARE USUALLY THE NUMBER OF VARIABLES IN THE FUNCTION (NVAR), MAX
c       AND MINIMUM STEP SIZE (MLIMIT,ACUR), NUMBER OF ITERATIONS FOR
c       THE PARTICULAR RUN (TIME), AND THE INITIAL GUESSES FOR THE
c       VARIABLES AND THEIR INITIAL STEP SIZE.
c
c

        REAL X(10),Dincrx(10),mlimit,mlimit10,acur,acur10

        Integer mdist,nvar,ihow,time

        open(unit=7,file='mainhc',status='new')

        write(5,*)'input # of variables in this calculation'
        read(6,*)nvar

c       write(5,*)'Input distance of climb in one direction (mdist)'
c       read(6,*)mdist

        mdist=10*nvar

        write(5,*)'Input starting values, and increments'
        write(7,*)'Initial Guess for Magnitude and Step Size'
        write(7,*)'  '

        do io=1,nvar

                write(5,*)'Please input guesses for variable #',io
                read(6,*)x(io)

                write(5,*)'input starting increments for x #',io
                read(6,*)dincrx(io)

                write(7,*)'Xi=',x(io),'Si=',dincrx(io),' for var. #',io

        end do

c
c       Input how many iterations for the particular run
c

        write(5,*)'How many iterations'
        read(6,*)time
        write(7,*)'Number of Iterations=',time
        write(7,*)'  '
c
c       Input the acurracy limit of this calculation
c

        write(5,*)' Input maximun step size'
        read(6,*)mlimit
```

```
write(7,*)'Maximum step size =',mlimit
write(7,*)' '

write(5,*)'Input the minimun step size'
Read(6,*)acur
write(7,*)'Minimun step size =',acur

write(7,*)'Variables move in same direction (mdist)',mdist

write(5,*)'Begin outputing HC data when?'
read(6,*)Ihow

CALL HC(nvar,x,time,dincrx,mdist,ihow,mlimit,acur)

close(7)
stop
end
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SUBROUTINE HC
C
C               THIS SUBROUTINE IS THE N-VARIABLE VERSION OF THE
C       THE HILL CLIMBER OPTIMIZATION ROUTINE DEVELOPED BY THE THESIS
C       STUDENT.  THIS IS THE MOST GENERAL PURPOSE ROUTINE AND IT
C       IS USED FOR TESTING THE OPTIMIZATION PROCESS ON GIVEN
C       QUADRATIC, LOGRYTHMIC, AND TRANSCENDENTAL
C       FUNCTIONS.
C
C
C

        SUBROUTINE HC(nvar,x,time,dincrx,mdist,ihow,mlimit,acur)


        REAL OERR,ERR,what,acur,acur10
        real mlimit,mlimit10,Merr(40000),Ediff
        real arrx(10,40000),x(10),dx(10),dincrx(10)
        real xsml(10),xlrg(10)
        real bestx(10)

        integer ch,wait,wait2,nvar,nvar2,ixdir(10),idch(10)
        integer te,td,time,mdist,ihow

C
C       INITIALIZATION OF COUNTERS
C

        ch=1
        oerr=1.0E30
        I=0
        K=0
        J=0
        L=0
        M=0

C
C       VARIABLES USED IN STEP SIZE RESTRAINT CODE
C

        mlimit10=mlimit*10.0
        acur10=acur/10.0

C
C       VARIABLES OF THE ROUTINE THAT ARE BASED ON THE NUMBER
C       OF VARIABLES BEING ADJUSTED BY THE PROCESS.
C


        nvar2=nvar*2
        wait=nvar*4
        wait2=nvar*2+1

c
c       INITIALIZATION OF A COUNTERS AND DIRECTION INDICATORS THAT DESCRIBE
C       THE PARTICULAR STATE OF EACH VARIABLE DURING THE OPTIMIZATION.
c
```

```
        do io=1,nvar

                dx(io)=1.0
                ixdir(io)=1
                xsml(io)=0.0
                xlrg(io)=0.0
                idch(io)=0

        end do

c
c       This begins the Hclimber by calling fuction to be optimized
c

 100    Call Func(err,x,nvar)

c
c       This is iteration counter used in Hclimber
c

        I=I+1

C
C       THE ARRAY MERR(I) CONTAINS ALL THE VALUES OF THE HILL HEIGHT.
C       EDIFF IS THE DIFFERENCE BETWEEN THE PRESENT HILL HEIGHT AND THE
C       PREVIOUS HILL HEIGHT AND OEDIFF IS THE DIFFERENCE BETWEEN THE
C       PRESENT HILL HEIGHT AND THE HILL HEIGHT OF LOWEST MAGNITUDE FOUND
C       TO DATE.
C

        Merr(I)=err
        Ediff=abs(Merr(I)-Merr(I-1))
        OEdiff=abs(err-oerr)

c
c       Printing option that prints out error,I, and Network prameters.
c

        if(I.ge.Ihow)then
        write(7,*)' '
        WRITE(7,15)I,err,ch,Ediff
 15     Format(/' I= ',I5,3x,'HH= ',E12.6,3x,'Ch=',I2,2x,'HHdif=',E13.6)
        write(7,*)' '
        write(7,*)'Magnitudes .............'
        write(7,17)x(1),x(2),x(3),x(4)
 17     Format(6x,'X1=',F12.6,2x,'x2=',F12.6,2x,'x3=',F12.6,2x,'x4=',F12.6)
        write(7,*)'  '

c       write(7,18)x(5),x(6),x(7),x(8)
c 18    Format(' X5= ',F16.6,3x,'x6=',F16.6,3x,'x7= ',F16.6,3x,'x8= ',F16.6)
c       write(7,*)x(9)
c       write(7,*)'  '

        write(7,*)' Step sizes ............'
        write(7,16)dincrx(1),dincrx(2),dincrx(3),dincrx(4)
  16    Format(6x,'s1=',F12.6,2x,'s2=',F12.6,2x,'s3=',F12.6,2x,'s4=',F12.6)
```

```
c       write(7,21)dincrx(5),dincrx(6),dincrx(7),dincrx(8)
c  21   Format('ix1=',F14.7,1x,'ix2=',F14.7,1x,'ix3=',F14.7,1x
c     $         ,'ix4=',F14.7)

        write(7,*)'  '
        write(7,*)' Direction .............'
        write(7,19)dx(1),dx(2),dx(3),dx(4)
  19    Format(6x,'d1=',F4.1,1x,'d2=',F4.1,1x,'d3=',F4.1,1x,'x4=',F4.1)
        write(7,*)'  '

        endif

c
c       This loop loads the storage array with all values of variables through
c       out the iteration sequence.
c

        do ix=1,nvar

                arrx(ix,I)=x(ix)

        end do

c
c       These two do loops determine determine when a variable needs to be
c       reduced in size when it has gotten relatively close to an answer
c       and begins jumping around it.
c

        IF(I.GE.wait)then

                do ia=1,nvar

C
C                       J,K,L,M AND MM ARE USED TO COUNT THE LENGTH OF
C                       A PERIODIC PATTERN.
C

                        j=1+1
                        k=1 + wait2
                        m=0
                        mm=0

                do ib=1,nvar2

C
C                       THIS STATEMENT DETECTS THE PERIODIC PATTERN.
C

                        if(arrx(ia,j).eq.arrx(ia,k))m=m+1
                        j=j+1
                        k=k+1

                end do

C
C               THIS STATEMENT PREVENTS PERIODIC PATTERN FROM BEING
C               DETECTED ONE RIGHT AFTER ANOTHER.
C
```

```
                if(1.le.idch(ia))goto 1
C
C       IF PATTERN IS DETECTED THEN THE STEP SIZE OF THE PERIODIC VARIABLE
C       IS REDUCED IN MAGNITUDE BY A FACTOR TEN AND THE MAGNITUDE OF
C       THE VARIABLE IS RESET TO THE BEST VALUE FOUND SO FAR IN THE PROCESS.
C
C
                if(m.eq.nvar2)then
                        idch(ia)=1 + wait2
                        x(ia)=bestx(ia)
                        dincrx(ia)=dincrx(ia)/10.0

c                       write(7,*)'***********--variable is',ia
c                       write(7,*)'Decreased increment of X,dincrx=',dincrx(ia)
c                       write(7,*)'X= ',X(ia)

                endif

   1    end do

        l=l+1

        endif

c
c       This loop keeps the step size of the variables from going out of limit.
c

        do if=1,nvar

        if(dincrx(if).le.acur10)then
                dincrx(if)=acur

c               write(7,*)'----------------------------------------'
c               write(7,*)'dincrxl has gone to low, increased it to',dincrxl

        endif

        end do

c
c       This block of code determines when to increases the size (mag.) of the
c       variable based on how far it has climber in one direction.
c

        do ic=1,nvar

C
C               THIS STATEMENT CHECKS THE VARIABLES DIRECTION INDICATOR
C               AND COUNTS HOW MANY ITERTION THEY HAVE MOVED IN THE SAME
C               DIRECTION.
C
                if(ixdir(ic).eq.1)then
                        xlrg(ic)=xlrg(ic)+1
                else
                        xlrg(ic)=0
                endif
```

```
C
C               THIS STATEMENT DOES THE SAME AS THE ABOVE FOR THE
C               OPPOSITE DIRECTION.
C

                if(ixdir(ic).eq.0)then
                        xsml(ic)=xsml(ic)+1
                else
                        xsml(ic)=0
                endif

C
C               THE NEXT TWO IF-THEN'S INCREASE THE STEP SIZE OF THE
C               VARIABLES IF THEY HAVW MOVED CONTINUOUSLY IN THE
C               SAME DIRECTION.
C

                if(xlrg(ic).eq.mdist)then
                        dincrx(ic)=dincrx(ic)*10.0

c                       Write(7,*)'############--variable is',ic
c                       Write(7,*)'Increased increment of X,dincrx=',dincrx(ic)

                        xlrg(ic)=0
                endif
c
c
                if(xsml(ic).eq.mdist)then
                        dincrx(ic)=dincrx(ic)*10.0

c                       write(7,*)'############ --variable is',ic
c                       write(7,*)'Increaesd increment of X,dincrx=',dincrx(ic)

                        xsml(ic)=0
                endif
        enddo

C
C       IF THE STEP SIZE HAS GONE ABOVE THE LIMIT SPECIFIED BY THE USER
C       THIS LOOP WILL REDUCE THE STEP SIZE BY A FACTOR OF TEN.
C

        do if=1,nvar

        if(dincrx(if).ge.mlimit10)then

                dincrx(if)=mlimit
c               write(7,*)'STEP SIZE GONE ABOVE LIMIT'
c               write(7,*)'  '
        endif

        end do

c
c       This block takes the first initial step in a positive incrmental
c       direction for X(1).
c

        IF(I.EQ.1)THEN
```

```
                x(1)=x(1) + dx(1)*dincrx(1)

                if(err.lt.oerr)oerr=err

                GOTO 100
        ENDIF

c
c       This block of code changes the variable back to its previous value
c       if changing it caused the hill height to increase.  This block also
c       adjusts the variables based on the direction indicators and step size
c       magnitudes by evaluating the magnitude of the hill height.
c


        do id=1,nvar

C
C               THIS OUTER IF-THEN STATEMENT PERFORMS THE VARIABLE ADJUSTMENTS
C               WHEN THE ROUTINE IS OPERATING ON THE LAST VARIABLE IN THE
C               FUNCTION.
C
                if(ch.eq.nvar)then

C
C               THIS IF-THEN REDUCES THE LAST VARIABLE BACK TO IT'S
C               PREVIOUS VALUE IF THE PRESENT HILL HEIGHT EQUALS THE
C               PREVIOUS HILL HEIGHT.  IT ALSO IMMEDIATELY ADJUSTS
C               THE FIRST VARIABLE TO ITS NEW POSITION.  THE STEP
C               SIZE OF THE LAST VARIABLE IS INCREASED BY A FACTOR
C               OF TEN.
C

                        IF(EDIFF.eq.0.0)THEN

                                x(nvar)=x(nvar)-dx(nvar)*dincrx(nvar)
                                x(1)=x(1)+dx(1)*dincrx(1)

                                DINCRX(NVAR)=DINCRX(NVAR)*10.0

c                       WRITE(7,*)'ERR  - PREVIOUS ERR LT 1E-6,Bx=',x(nvar)
c                       WRITE(7,*)'I=',I,' id=',id,' step=',dincrx(nvar)

                        goto 70
                        ENDIF

C
C                       THE SAME AS ABOVE IS PERFORMED WHEN THE PRESENT
C                       HILL HEIGHT EQUALS THE BEST HILL HEIGHT FOUND
C                       TO DATE.
C

                        IF(OEDIFF.eq.0.0)THEN

                                x(nvar)=x(nvar)-dx(nvar)*dincrx(nvar)
                                x(1)=x(1)+dx(1)*dincrx(1)
                                DINCRX(NVAR)=DINCRX(NVAR)*10.0

c                               WRITE(7,*)'Oerr and Err are = to e-6'
c                               write(7,*)'EEEERRRRRRRRRRR = OOOOeeerrrr'
```

```
c                              WRITE(7,*)'I=',I,' id=',id,' step=',dincrx(nvar)

                               goto 70

                         ENDIF

C
C                        IF THE HILL HEIGHT IS LOWER THEN THE LAST VARIABLE
C                        IS KEPT AT IT'S NEW MAGNITUDE AND THE FIST VARIABLE
C                        IS ADJUSTED.
C

                         if(err.lt.oerr)x(1)=x(1)+dx(1)*dincrx(1)

C
C                        IF THE HILL HEIGHT IS GREATER THE LAST VARIABLE
C                        IS REDUCED AND FIRST VARIABLE IS ADJUSTED IMMEDIATELY.
C

                         if(err.gt.oerr)then
                               x(nvar)=x(nvar)-dx(nvar)*dincrx(nvar)
                               x(1)=x(1)+dx(1)*dincrx(1)
                         endif
                    goto 70
                    endif

C
C                   THE FOLLOWING IF-THEN STATEMENTS ARE FOR ALL OF THE ITERATIONS
C                   BESIDES ADJUSTMENT OF THE LAST VARIABLE ( IN OTHER WORDS
C                   MOVEMENT OF THE VARIABLES IN SEQUENCE 1,2,3,..... N, BUT
C                   NOT N.  MOVEMENT OF THE LAST VARIABLE (N) IS HANDLED ABOVE.
C                   CH TRACKS WHICH VARIABLE IS CURRENTLY BEING ADJUSTED.
C

                    IF(CH.EQ.ID .AND. EDIFF.eq.0.0)THEN

                         TE=ID+1
                         x(id)=x(id)-dx(id)*dincrx(id)
                         x(te)=x(te)+dx(te)*dincrx(te)
                         DINCRX(ID)=DINCRX(ID)*10.0

c                        WRITE(7,*)'ERR  - PREVIOUS ERR LT 1E-6,Bx=',x(id)
c                        WRITE(7,*)'I=',I,' id=',id,' step=',dincrx(nvar)

                    goto 70

                    ENDIF


                    IF(CH.EQ.ID .AND. OEDIFF.eq.0.0)THEN

                         TE=ID+1
                         x(id)=x(id)-dx(id)*dincrx(id)
                         x(te)=x(te)+dx(te)*dincrx(te)
                         DINCRX(ID)=DINCRX(ID)*10.0

c                        WRITE(7,*)'Oerr and Err are = to e-6'
c                        write(7,*)'EEEERRRRRRRRRRR = OOOOeeerrrr'
c                        WRITE(7,*)'I=',I,' id=',id,' step=',dincrx(nvar)
```

```
                        goto 70
                ENDIF


                IF(ch.eq.id .and. err.lt.oerr)then

                        td=id+1
c                       if(id.eq.nvar)td=1
                        x(td)=x(td)+dx(td)*dincrx(td)

                endif


                If(ch.eq.id .and. err.gt.oerr)then
                        td=id+1
                        x(id)=x(id)-dx(id)*dincrx(id)
                        x(td)=x(td)+dx(td)*dincrx(td)

                endif

        end do

c
c       This loop changes the variables direction based on the magnitude of
c       the hill height.  If the variables adjustment causes the hill height
c       not to be reduced below the current best hill height then the
c       particular variables direction indicators are reversed so the variable
c       can move in the other direction.
c


 70     do ie=1,nvar

                if(ch.eq.ie .and. ixdir(ie).eq.1 .and. err.gt.oerr)then
                        ixdir(ie)=0
                        dx(ie)=-1.0
                        goto 20
                endif

                if(ch.eq.ie .and. ixdir(ie).eq.0 .and. err.gt.oerr)then
                        ixdir(ie)=1
                        dx(ie)=1.0
                        goto 20
                endif

        end do


c
c       This block keeps track of which variable is to be changed next.
c


 20     if(ch.le.nvar)then
                if(ch.eq.nvar)ch=0
                ch=ch+1
        endif

c
```

```
c       This block records value of the best variables and current values
c       as the Hclimber proceeds thru the climb.
c

        if(err.lt.oerr)then

                write(5,*)'err=',err,' at I=',I
                oerr=err

                do ip=1,nvar
                        bestx(ip)=arrx(ip,I)
                end do

        endif

c
c       This tells machine to do another iteration
c

        IF(I.LT.time)GOTO 100
        write(7,*)' END OF RUN'
        write(7,*)'  '
        write(7,*)'Values of the Variables corresponding to HHltd'
        write(7,*)'X1=',bestx(1),'X2=',bestx(2),'B3=',bestx(3)
        write(7,*)'Hill Height =',oerr

        return
        end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C
C       SUBROUTINE FUNC
C
C               THIS SUBROUTINE IS CALLED BY THE HILL CLIMBER OPTIMIZATION
C       ROUTINE.  IT CALCULATES THE MAGNITUDE OF THE HILL HEIGHT BASED
C       ON WHATREVER FUNCTION IT CONTAINS.  THE FUNCTIONS  CONTAINED
C       IN THIS SUBROUTINE ARE USE TO TEST THE PERFORMANCE OF THE
C       THE OPTIMIZATION ROUTINE DEVELOPED BY THE THESIS STUDENT.
C

        subroutine func(err,x,nvar)

        integer nvar
        real x(nvar),err
        pi=3.141593
        q=x(1)

        err= (x(2)-20.0*SIN(0.05*x(1)))**2 + 0.1*(x(2)**2 + x(1)**2)

c       err=(exp(-x(1))-10.0*cos(q)+0.5*sin(q)-X(2)**2.+exp(-x(2)))**2.

c       err=(q**5-4.5*q**4+4.55*q**3+2.675*q**2-3.3*q-1.4375-x(2))**2

c       err=100*(x(2) - x(1)**2)**2 + (1-x(1))**2

c       err=(x(1)**2 - 78.7)**2

c       ERR = ERR1 + ERR2 + ERR3 + ERR4 + ERR5 + ERR6

        return
        end


c
```

## APPENDIX E

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       PROGRAM V6PATT
c
c             THIS PROGRAM PERFORMS THE HILL CLIMBER OPTIMIZATION
c       ON THE SIX CAPACITIVE PARAMETERS OF THE PI NETWORKS IN AN ATTEMPT
c       TO ENFORCE THE DESIRED CURRENT DISTRIBUTION ON THE ARRAY ELEMENTS.
c       THIS PROGRAM CALLS CAP6HC.FOR WHICH IS AN ADAPTED HILL CLIMBER
c       OPTIMIZATION SUBROUTINE THAT HAS BE MODIFIED TO ANALYSIS THE
c       EQUILATERAL ARRAY DESIGN PROBLEMS.  THE NECESSARY INPUTS FROM
c       THE USER ARE ENTER IN THIS PROGRAM AND PLOTTING OF THE RESULTING
c       RADIATION PATTERN CAN BE DONE BY CALLING SUBROUTINE PATT3.
c
c
c
c

        REAL Tlght,OERR,ERR,length,kl,what,SWR,rt,Mit,phit,Pin,pi,acur,acur10
        real scale,ind1,ind2,ind3,mlimit,mlimit10,max,null
        real x(10),dincrx(10)
        real FAMP(4),FPH(4),Cap(10)
        real bestx(10),amp(3),ph(3),ampC(3),phC(3),L1,L2,L3


        complex Z11,Z12,Z13,It,I1,I2,I3,ckt1(3),ckt2(3),ckt3(3),B(15)
        complex S1(4),w(3),volt1,Zin,Rfcoeff,cj,I123(3)
        COMPLEX B123(3)

        integer start,often,ch,wait,wait2,nvar,nvar2,ixdir(10),idch(10),num1
        integer te,td,time,mdist,ihow,icount,dir(10,40000)

        open(unit=7,file='v6PATT_one',status='new')

c
c       Self and Mutual impedances values for the 3 elements
c
        z11=(36.5,21.0)
        z12=(20.4,-14.18)
        z13=(20.4,-14.18)

        pi=2.0*asin(1.0)
        cj=(0.0,1.0)

c
c       These vars. define the dimensions of the 3 element triangular array
c
c       if(Tlght.eq.0.125)length=4.0
c       if(Tlght.eq.0.25)length=2.0
c       if(Tlght.eq.0.5)length=1.0

        length=4.0

c       write(5,*)'Input lenght of T.L.'
c       read(6,*)Tlght

        TLGHT=.125


c       write(5,*)' Given Inductor Values (ohms).'
c       read(6,*)ind1,ind2,ind3
```

```
           IND1=10.0
           IND2=15.0
           IND3=22.0

           write(7,*)'The Inductive Reactances (ohms).'
           write(7,*)'XL13=',IND1,' XL12=',IND2,' XL23=',IND3
           write(7,*)'  '

           ckt1(1)=cmplx(0.0,ind1)
           ckt2(1)=cmplx(0.0,ind2)
           ckt3(1)=cmplx(0.0,ind3)

           L1=IND1/(2.0*PI*7.0E6)
           L2=IND2/(2.0*PI*7.0E6)
           L3=IND3/(2.0*PI*7.0E6)

           WRITE(7,*)'At 7 MHz the Inductor Values are (henries),'
           WRITE(7,*)'L1=',L1,' L2=',L2,' L3=',L3
           write(7,*)

           write(7,*)'Initial guess and step size for the Caps ',ip
           write(7,*)'    '

           do ip=1,6

           write(5,*)'Initial guess and step size for Cap #',ip
           read(6,*)X(ip),dincrx(ip)
           write(7,*)'For Cap#',ip,'Xi=',x(ip),'Si=',dincrx(ip)

           end do

           write(7,*)'  '
           WRITE(5,*)'INPUT LENGTH(# OF ITERATIONS)?'
           READ(6,*)TIME
           WRITE(7,*)'Number of Iterations =',TIME

           WRITE(5,*)'MAX STEP SIZE?'
           READ(6,*)MLIMIT
           write(7,*)' '
           write(7,*)'MAXIMUM STEP SIZE?',mlimit

           write(5,*)'MINIMUM STEP SIZE (acurracy)?'
           Read(6,*)acur
           WRITE(7,*)' '
           WRITE(7,*)'MINIMUM STEP IS ',ACUR

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c          These are the input/given current distribution on 3 elements
c
           write(7,*)' '
           write(7,*)'!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!'
           do ir=1,3

                   write(5,*)'Input Magnitude and Phase'
                   read(6,*)amp(ir),ph(ir)

           end do
```

```
        write(7,*)'Input Om and On'
        read(6,*)max,null


        WRITE(5,*)' MAX=',MAX,' NULL ',NULL
        write(7,*)'    '
        Write(7,*)'Current Distribution Corresponds to a Om and On of'

        WRITE(7,*)'Om =',MAX,' On =',NULL

c
c       Change amp and phase to real and imaginary component
c

        CALL CMPN(3,AMP,PH,W)

c
c       Change complex current values to normal amplitudes and phase
c

        CALL MAG_PH_NORM(3,W,AMP,PH)


        CALL CMPN(3,AMP,PH,W)

        write(7,*)'    '
        write(7,*)'Amps Normalized =',amp(1),amp(2),amp(3)
        write(7,*)'Phase (Deg.) =',ph(1)*180.0/pi,ph(2)*180.0/pi,ph(3)*180./pi


c
c       Calculate It (input current into array) using normalized given curr.
c
        write(7,*)'  '
        It=(w(1)*Z11 + w(2)*Z12 + w(3)*Z13)/(50.0,0.0)
        write(7,*)'It (Desired input current) =',It



        CALL CAP6HC(ckt1,ckt2,ckt3,PI,W,IT,X,DINCRX,TIME,MLIMIT,ACUR,S1)


        CALL MAG_PH_NORM(4,S1,FAMP,FPH)

        rt=180.0/pi

        write(5,*)' I=',I
        write(7,*)' I=',I
        write(7,*)'  '
        write(7,*)'AMPS=',Famp(1),Famp(2),Famp(3),'ITM=',FAMP(4)
        write(7,*)'PHASE=',Fph(1)*rt,Fph(2)*rt,Fph(3)*rt,'ITPH=',FPH(4)*RT
        write(7,*)'  '
        write(7,*)'Best Capacitative Reactances'

        write(7,*)x(1),x(2),x(3),x(4)
        write(7,*)x(5),x(6)
        WRITE(7,*)' '
        write(7,*)'Capacitance of elements at 7 MHz'
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       Calculates the Capacitance of L.C. Netwirk Parameters
c

        do ip=1,nva.

        cap(ip)=1.0/(x(ip)*2.0*pi*7.0E6)
        write(7,*)' Cap. of ',ip,' is',cap(ip),' Farads'

        end do


        voltl=S1(1)*z11 + S1(2)*z12 + S1(3)*z13

        Zin=voltl/S1(4)

        Rfcoeff=(Zin-(50.0,0.0))/(Zin+(50.0,0.0))
        SWR=(1+cabs(Rfcoeff))/(1.0 - cabs(rfcoeff))

        write(7,*)'  '
        write(7,*)'Zin=',Zin,' SWR=',SWR
        write(7,*)' Itbest=',S1(4),'V1=',voltl

        write(5,*)'Zin found=',Zin
        write(5,*)' SWR=',SWR
c
c       Calculate values for It for singlr dipole problem
c

        Pin=(((CABS(S1(4)))**2.0)*REAL(ZIN))

        scale=sqrt(1000.0/Pin)

        mit=sqrt((1000.0)/36.5)

        phit=0.0


c       numl=1
c       call patt3(scale,Famp,Fph,pi,length,numl,mit,phit)

        numl=2
        Famp(1)=1.0

        do ip=1,3
                Famp(ip)=Famp(ip)*scale
        end do

        call patt3(scale,Famp,Fph,pi,length,numl,mit,phit)


        CLOSE(7)
        STOP
        END

        INCLUDE 'CAP6HC.FOR'
        INCLUDE 'FUNC6C.FOR'
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c
c
c
c
        SUBROUTINE CAP6HC(cktl,ckt2,ckt3,pi,w,It,x,dincrx,time,mlimit,acur,S1)


        REAL OERR,ERR,what,acur,acur10
        real mlimit,mlimit10,Merr(40000),Ediff
        real arrx(10,40000),x(10),dx(10),dincrx(10)
        real xsml(10),xlrg(10)
        real bestx(10),pi,BSTIN(3)

        complex B(15),It,w(3),S1(4),cktl(3),ckt2(3),ckt3(3)

        integer ch,wait,wait2,nvar,nvar2,ixdir(10),idch(10)

        integer te,td,time,mdist,ihow

        IHOW=time-1


        ch=1
        oerr=1.0E30
        I=0
        K=0
        J=0
        L=0
        M=0

        mlimit10=mlimit*10.0
        acur10=acur/10.0


        nvar=6
        mdist=nvar*10
        nvar2=nvar*2
        wait=nvar*4
        wait2=nvar*2+1

c
c       Initialize points for Hclimber vars. and increments
c

        do io=1,nvar

                dx(io)=1.0
                ixdir(io)=1
                xsml(io)=0.0
                xlrg(io)=0.0
                idch(io)=0

        end do

c
```

```
c       This begins the Hclimber by calling fuction to be optimized
c


 100    Call Func6c(pi,w,x,it,b,cktl,ckt2,ckt3,err,S1)


c
c       This is iteration counter used in Hclimber
c

        I=I+1
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       This block takes the first initial step in a positive incrmental
c       direction for X(1).
c
        IF(I.EQ.1)THEN
                x(1)=x(1) + dx(1)*dincrx(1)
                write(7,*)'  '
                WRITE(7,*)'Initial Hill Height =',err
                if(err.lt.oerr)then
                        oerr=err
                endif
                GOTO 100
        ENDIF

cccccccccccccccccccccccccccccccccccccccccccccccccc

        Merr(I)=err
        Ediff=abs(Merr(I)-Merr(I-1))
        OEdiff=abs(err-oerr)

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c
c       Printing option that prints out error,I, and Ntwk. elements as desired
c
        if(I.gt.Ihow)then
        write(7,*)' '
        write(7,*)'Number of Iterations,HHi,Variable Seq., & HH(i)-HH(i-1)'
        write(7,*)
        WRITE(7,15)I,err,ch,ediff
 15     Format(/' I= ',I5,2x,'HH= ',E12.6,2x,'Ch=',I2,2x,'HHdif=',E12.6)
        write(7,*)'Varibles present value ....'
        write(7,*)'  '
        write(7,17)x(1),x(2),x(3),x(4)
 17     Format(6x,'X1=',F12.6,2x,'x2=',F12.6,2x,'x3=',F12.6,2x,'x4=',F12.6)
        write(7,*)'  '
        write(7,18)x(5),x(6)
 18     Format(6x,'X5=',F12.6,2x,'x6=',F12.6)

        write(7,*)'  '
        write(7,*)' Step sizes ...........'
        write(7,*)'   '

        write(7,16)dincrx(1),dincrx(2),dincrx(3),dincrx(4)
  16    Format('s1=',F12.6,1x,'s2=',F12.6,1x,'s3=',F12.6,1x
     $  ,'s4=',F12.6)
```

```
        write(7,21)dincrx(5),dincrx(6)
  21    Format('s5=',F12.6,1x,'s6=',F12.6)

        write(7,*)'  '
        write(7,*)' Direction ............'
        write(7,*)'   '
        write(7,19)dx(1),dx(2),dx(3),dx(4)
  19    Format('d1=',F4.1,1x,'d2=',F4.1,1x,'d3=',F4.1,1x,'x4=',F4.1)

        write(7,*)'  '

        endif

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c
c       This loop loads the storage array with all values of variables through
c       out the iteration sequence.
c

        do ix=1,nvar

                arrx(ix,I)=x(ix)

        end do

c
c       These two do loops determine determine when a variable needs to be
c       reduced in size when it has gotten relatively close to an answer
c       and begins jumping around it.
c

        IF(I.GE.wait)then

                do 1 ia=1,nvar

                        j=1+1
                        k=1 + wait2
                        m=0
                        mm=0

                do 2 ib=1,nvar2

                        if(arrx(ia,j).eq.arrx(ia,k))m=m+1
                        j=j+1
                        k=k+1

  2             continue

                if(1.le.idch(ia))goto 1

                if(m.eq.nvar2)then
                        idch(ia)=1 + wait2
                        x(ia)=bestx(ia)
                        dincrx(ia)=dincrx(ia)/10.0
c                       write(7,*)'***********--variable is',ia
c                       write(7,*)'Decreased increment of X,dincrx=',dincrx(ia)
c                       write(7,*)'X= ',X(ia)
```

```
                endif

   1      continue
          l=l+1
          endif



  c
  c       This loop keeps the acurracy of the variables from going out o: limit
  c


          do if=1,nvar

          if(dincrx(if).le.acur10)then
                  dincrx(if)=acur
  c               write(7,*)'-----------------------------------------'
  c               write(7,*)'dincrxl has gone to low, increased it to',dincrxl
          endif

          end do

  c
  c       This block of code determines when to increases the size (mag.) of the
  c       variable based on how far it has climber in one direction.
  c

          do 3 ic=1,nvar

                  if(ixdir(ic).eq.1)then
                          xlrg(ic)=xlrg(ic)+1
                  else
                          xlrg(ic)=0
                  endif
  c
  c
                  if(ixdir(ic).eq.0)then
                          xsml(ic)=xsml(ic)+1
                  else
                          xsml(ic)=0
                  endif

                  if(xlrg(ic).eq.mdist)then
                          dincrx(ic)=dincrx(ic)*10.0

  c                       Write(7,*)'############--variable is',ic
  c                       Write(7,*)'Increased increment of X,dincrx=',dincrx(ic)

                          xlrg(ic)=0
                  endif
  c
  c
                  if(xsml(ic).eq.mdist)then
                          dincrx(ic)=dincrx(ic)*10.0

  c                       write(7,*)'############ --variable is',ic
  c                       write(7,*)'Increaesd increment of X,dincrx=',dincrx(ic)
```

```
                  xsml(ic)=0
               endif
  3         continue

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

            do if=1,nvar

            if(dincrx(if).ge.mlimit10)then

                  dincrx(if)=mlimit
c                 write(7,*)'STEP SIZE GONE ABOVE LIMIT'
c                 write(7,*)'  '
            endif

            end do

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c           This block of code changes the variable back to its previous value
c           if changing it caused the error to increase.  This block also increases
c           or decreases the variable based on the current direction of the var., by
c           the value in incrx(?).
c


            do 4 id=1,nvar

                  if(ch.eq.nvar)then

                        IF(EDIFF.eq.0.0)THEN

                              x(nvar)=x(nvar)-dx(nvar)*dincrx(nvar)
                              x(1)=x(1)+dx(1)*dincrx(1)

                              DINCRX(NVAR)=DINCRX(NVAR)*10.0

c                       WRITE(7,*)'ERR  - PREVIOUS ERR LT 1E-6,Bx=',x(nvar)
c                       WRITE(7,*)'I=',I,' id=',id,' step=',dincrx(nvar)

                        goto 70
                        ENDIF


                        IF(OEDIFF.eq.0.0)THEN

                              x(nvar)=x(nvar)-dx(nvar)*dincrx(nvar)
                              x(1)=x(1)+dx(1)*dincrx(1)
                              DINCRX(NVAR)=DINCRX(NVAR)*10.0

c                             WRITE(7,*)'Oerr and Err are = to e-6'
c                             write(7,*)'EEEERRRRRRRRRRR = OOOOeeerrrr'
c                             WRITE(7,*)'I=',I,' id=',id,' step=',dincrx(nvar)

                              goto 70

                        ENDIF
                  if(err.lt.oerr)x(1)=x(1)+dx(1)*dincrx(1)
                  if(err.gt.oerr)then
                         x(nvar)=x(nvar)-dx(nvar)*dincrx(nvar)
                         x(1)=x(_)+dx(1)*dincrx(1)
                  endif
           goto 70
           endif

           IF(CH.EQ.ID .AND. EDIFF.eq.0.0)THEN

                  TE=ID+1
                  x(id)=x(id)-dx(id)*dincrx(id)
                  x(te)=x(te)+dx(te)*dincrx(te)
                  DINCRX(ID)=DINCRX(ID)*10.0

c                 WRITE(7,*)'ERR  - PREVIOUS ERR LT 1E-6,Bx=',x(id)
c                 WRITE(7,*)'I=',I,' id=',id,' step=',dincrx(nvar)

           goto 70

           ENDIF



           IF(CH.EQ.ID .AND. OEDIFF.eq.0.0)THEN

                  TE=ID+1
                  x(id)=x(id)-dx(id)*dincrx(id)
                  x(te)=x(te)+dx(te)*dincrx(te)
                  DINCRX(ID)=DINCRX(ID)*10.0

c                 WRITE(7,*)'Oerr and Err are = to e-6'
c                 write(7,*)'EEEERRRRRRRRRRRR = OOOOeeerrrr'
c                 WRITE(7,*)'I=',I,' id=',id,' step=',dincrx(nvar)

                  goto 70
           ENDIF



           IF(ch.eq.id .and. err.lt.oerr)then

                  td=id+1
c                 if(id.eq.nvar)td=1
                  x(td)=x(td)+dx(td)*dincrx(td)

           endif


           If(ch.eq.id .and. err.gt.oerr)then
                  td=id+1
                  x(id)=x(id)-dx(id)*dincrx(id)
                  x(td)=x(td)+dx(td)*dincrx(td)

           endif
```

```
  4     continue
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       This loop changes the variables direction basd on the value of error
c


 70     do 5 ie=1,nvar

        if(ch.eq.ie .and. ixdir(ie).eq.1 .and. err.gt.oerr)then
                ixdir(ie)=0
                dx(ie)=-1.0
                goto 20
        endif
c
c
        if(ch.eq.ie .and. ixdir(ie).eq.0 .and. err.gt.oerr)then
                ixdir(ie)=1
                dx(ie)=1.0
                goto 20
        endif

  5     continue
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c
c       This block keeps track of which variable is to be changed next.
c

 20     if(ch.le.nvar)then
                if(ch.eq.nvar)ch=0
                ch=ch+1
        endif

c
c       This block records value of the best variables and current values
c       as the Hclimber proceeds thru the climb.
c

        if(err.lt.oerr)then

                oerr=err

                do ip=1,nvar
                        bestx(ip)=arrx(ip,I)
                end do
        endif

c
c       This tells machine to do another iteration
c

        IF(I.LT.time)GOTO 100

        do iz=1,nvar

                x(iz)=bestx(iz)
```

```
end do

write(7,*)' END OF RUN'
'rite(7,*)'  '
write(7,*)'Lowest Hill Height Magnitude =',oerr
write(7,*)'   '

return
end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       SUBROUTINE FUNC6C
c
c               THIS SUBROUTINE PERFORMS THE ANALYSIS OF THE EQUILATERAL
c       ARRAY BY CALLING THE ANLYSUBR SUBROUTINE.  THE SIX CAPACITIVE
c       REACTANCES BEING ADJUSTED BY THE HILL CLIMBER OPTIMIZATION
c       ROUTINE ARE SENT TO ANLYSUBR FOR CALCULATION OF THE RESULTING
c       CURRENT DISTRIBUTION.  THE HILL HEIGHT IS THEN CALCULATED AND
c       RETURNED TO THE OPTIMIZATION ROUTINE FOR EVALUATION.  THE DESIRED
c       CURRENTS ARE INPUT AS VARIABLE W AND THE PI NETWORK PARAMETERS
c       ARE IN VECTORS CKT1,CKT2,CKT3.
c
c       INPUTS:  PI,W,X,IT,CKT1(1),CKT2(1),CKT3(1)
c       OUTPUT:  ERR,S1,B
c
c

        SUBROUTINE FUNC6C(PI,W,X,IT,B,CKT1,CKT2,CKT3,ERR,S1)

        REAL err,x(10),Pi,AMPCN(4),PHCN(4),Tlght

        COMPLEX ckt1(3),ckt2(3),ckt3(3),B(15),s1(4),w(3),It,B123(4)

c
c       THIS VARIABLE IS THE LENGTH OF THE TRANSMISSION LINE
c       BETWEEN THE ARRAY ELEMENTS AND THE PI NETWORKS.
c

        TLGHT=.125

c
c       THE PRESENT VALUES FOR THE SIX CAPACITIVE REACTANCES ARE LOADED
c       INTO THE VARIABLES TO BE PASSED INTO THE ANALYSIS ROUTINE.
c

        ckt1(2)=cmplx(0.0,x(1))
        ckt2(2)=cmplx(0.0,x(2))
        ckt3(2)=cmplx(0.0,x(3))

        ckt1(3)=cmplx(0.0,x(4))
        ckt2(3)=cmplx(0.0,x(5))
        ckt3(3)=cmplx(0.0,x(6))


c
c       This is a call to subr. curr in file anlysubr.for which solves
c       the analysis of the array.
c

        call curr(pi,Tlght,ckt1,ckt2,ckt3,it,b)

c
c       This loops detects when the variables have gone below zero and
c       adds the square of the variables magnitude to the hill height.
c

c       err8=0.0
c       do ip=1,nvar
c
```

```
c       if(x(ip).lt.0.0)then
c
c               err8=err8 + x(ip)**2
c               write(7,*)'x=',ip,' gone LT. 0.0',x(ip)
c
c       endif
c       end do

c
c       This loop is a warning block of code which detects when the
c       currents for the array elements found by the analysis routine
c       are zero (undetermined).  This means that there is no solution
c       for the present values of the network parameters.
c
c

        do ip=1,3

                if(real(b(ip)).eq.0.0 .and. aimag(b(ip)).eq.0.0)then
                        write(7,*)'   '
                        write(7,*)'WARNING CURR',ip,' .... to ZEROES'
                        write(5,*)'WARNING CURR',ip,' .... to ZEROES'
                        write(7,*)'   '
                endif

                b123(ip)=b(ip)

        end do

        b123(4)=b(1)+b(4)+b(6)

c
c       Changes complex current distributions in normalized magnitude
c       and phase vectors.
c

        CALL MAG_PH_NORM(4,B123,AMPCN,PHCN)

c
c       Changes normalized magnitude and phase vectors into complex values.
c

        CALL CMPN(4,AMPCN,PHCN,S1)

c
c       This is the calculation of the hill height, which is the
c       difference between the desired and calculated values.
c

        err1=(real(w(1))-real(s1(1)))**2.+(real(w(2))-real(s1(2)))**2.
        err2=(real(w(3))-real(s1(3)))**2.+(aimag(w(1))-aimag(s1(1)))**2.
        err3=(aimag(w(2))-aimag(s1(2)))**2.
        err4=(aimag(w(3))-aimag(s1(3)))**2.

        err=err1+err2+err3+err4 + err8

        RETURN
        END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SUBROUTINE MAG_PH_NORM
C
C       This subroutine is used to convert a complex number
C       into Phasor form (magnitude and phase).  Once in
C       phasor form the magnitude and phases are then normalized
C       with respect to the current flowing on the first antenna.
C       That is the first complex number in cmplxI which is cmplxI(1).
C
C
C       Input : V,CMPLXI
C
C       Output : MAGTN,PHSEN
C
C

        SUBROUTINE MAG_PH_NORM(V,CMPLXI,MAGTN,PHSEN)

        INTEGER A,V
        REAL MAGTN(V),PHSEN(V)
        COMPLEX CMPLXI(V)

C
C       CALCULATIONS OF THE MAGNITUDE AND PHASE OF A INPUT COMPLEX NUMBER.
C

        DO A=1,V

                MAGTN(A)=CABS(CMPLXI(A))
                PHSEN(A)=ATAN2(AIMAG(CMPLXI(A)),REAL(CMPLXI(A)))

        END DO

C
C       NORMALIZATION ACCOMPLISHED BY DIVISION OF THE MAGNITUDES
C       AND SUBTRACTION OF THE PHASES.
C

        DO A=2,V

                MAGTN(A)=MAGTN(A)/MAGTN(1)
                PHSEN(A)=PHSEN(A)-PHSEN(1)

        END DO

        MAGTN(1)=1.0
        PHSEN(1)=0.0

        RETURN
        END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SUBROUTINE CMPN
C
C
C       THIS SUBROUTINE WAS DESIGNED TO CALCULATED THE COMPLEX
C       VALUE FROM INPUT MAGNITUDE AND PHASE VECTORS.
C       GIVEN THE MAGNITUDE AND PHASE OF A VECTOR THE COMPLEX
C       NUMBER REPRESENTATION IS FOUND USING THE FORMULA
C
C               M*COS(0) + j*M*SIN(0)
C
C       INPUTS : V,MAGN,PHSE
C       OUTPUT : CMPLXN
C
C


        SUBROUTINE CMPN(V,MAGN,PHSE,CMPLXN)

        INTEGER A,V
        REAL MAGN(V),PHSE(V)
        COMPLEX CMPLXN(V)

        DO A=1,V

        CMPLXN(A)=MAGN(A)*COS(PHSE(A)) + (0.0,1.0)*MAGN(A)*SIN(PHSE(A))

        END DO

        RETURN
        END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SUBROUTINE CURR
C
C               THIS SUBROUTINE PERFORMS THE EVALUATION
C       OF THE 15 LINEAR COMPLEX EQUATIONS THAT DESRCIBE THE
C       EQUILATERAL THREE ELEMENT ARRAY.  THE PI NETWORK PARAMETERS
C       (3 CAPACITORS AND 6 INDUCTORS) ARE INPUT INTO THIS ROUTINE
C       AND ARE CONSTANTS PROVIDED BY THE USER.  THE PARTICULAR
C       MAGNITUDE OF THESE PARAMETERS AND THE LENGTH OF THE
C       TRANSMISSION LINES ARE WHAT DETERMINE THE COEFFICENTS
C       OF THE 15 UNKNOWN VARIABLES.  THIS ROUTINE USES A LINEAR
C       EQUATION SOLVER ACCESSED FROM THE IMSL LIBRARY OF FORTRAN
C       SUBROUTINES.
C
C
C


        subroutine CURR(pi,Tlght,cktl,CKT2,CKT3,It,B)

c
c       ------- declare all variables----------
c

        INTEGER N,IA,M,IB,IJOB,IER
        COMPLEX A(15,15),B(15),WA(255),CKT1(3),CKT2(3),CKT3(3),XL13,XC13
        COMPLEX XC31,Z11,Z12,Z13,IT,C46,C02,PII,X1,X2,X3,X4,X5,X6
        REAL wk(15),Tlght,pi,L46,L02,LII

c
c       -------matrix parameters------------
c

        IA=15
        IB=15
        N=15
        M=1

C
C       ------mutual impedances-------------
C

        Z11=(36.5,21.0)
        Z13=(20.4,-14.18)
        Z12=Z13

C
C       ------calculation of constants based on transmission line-------
C

        L46=50.0*(sin(3.0*pi*Tlght))
        L02=(sin(3.0*pi*Tlght))/50.0
        LII=cos(3.0*pi*Tlght)

        C46=cmplx(0.0,L46)
        C02=cmplx(0.0,L02)
        PII=cmplx(LII,0.0)

c
```

```
c       ----set constant matrix b back to zero----
c
        do j=1,15
                B(j)=(0.0,0.0)
        end do
C
C       ------DECLARE INPUT CURRENT IT--------
C
        B(1)=IT
C
C       ------BEGIN CALCULATING THE MATRIX "A" COEFFS-------
C
        XL13=CKT1(1)
        XC13=CKT1(2)
        XC31=CKT1(3)
C
C       CALL SUBROUTINE COEFF FOR FIRST TIME
C
        CALL COEFF(C46,C02,PII,XL13,XC13,XC31,X1,X2,X3,X4,X5,X6)
C
C       ------INSERT COEFFS INTO MATRIX "A"-------
C
        A(1,1)=(1.,0.)
        A(1,4)=(1.,0.)
        A(1,6)=(1.,0.)
        A(2,2)=(1.,0.)
        A(2,5)=(1.,0.)
        A(2,8)=(1.,0.)
        A(3,3)=(1.,0.)
        A(3,7)=(1.,0.)
        A(3,9)=(1.,0.)
C
C
        A(4,1)=Z11
        A(4,2)=Z12
        A(4,3)=Z13
        A(4,10)=X1
        A(4,11)=X2
C
C
        A(5,6)=(1.,0.)
        A(5,10)=X3
        A(5,11)=X4
C
C
        A(6,1)=Z12
        A(6,2)=Z13
        A(6,3)=Z11
        A(6,10)=X2
        A(6,11)=X5
C
```

```
C
      A(7,7)=(1.,0.)
      A(7,10)=X4
      A(7,11)=X6

C
C     -------SECOND SET OF EQUATIONS------
C

      XL13=CKT2(1)
      XC13=CKT2(2)
      XC31=CKT2(3)
C
C

      CALL COEFF(C46,C02,PII,XL13,XC13,XC31,X1,X2,X3,X4,X5,X6)

C
C     ------LOAD COEFFS INTO MATRIX "A"-----
C
      A(8,1)=Z11
      A(8,2)=Z12
      A(8,3)=Z13
      A(8,12)=X1
      A(8,13)=X2
C
C
      A(9,4)=(1.,0.)
      A(9,12)=X3
      A(9,13)=X4
C
C
      A(10,1)=Z12
      A(10,2)=Z11
      A(10,3)=Z13
      A(10,12)=X2
      A(10,13)=X5
C
C
      A(11,5)=(1.,0.)
      A(11,12)=X4
      A(11,13)=X6

C
C     -----THIRD SET OF EQUATIONS------
C

      XL13=CKT3(1)
      XC13=CKT3(2)
      XC31=CKT3(3)

C
C
      CALL COEFF(C46,C02,PII,XL13,XC13,XC31,X1,X2,X3,X4,X5,X6)
C
C
      A(12,1)=Z12
      A(12,2)=Z11
      A(12,3)=Z13
      A(12,14)=X1
```

```
        A(12,15)=X2
C
C
        A(13,8)=(1.,0.)
        A(13,14)=X3
        A(13,15)=X4
C
C
        A(14,1)=Z12
        A(14,2)=Z13
        A(14,3)=Z11
        A(14,14)=X2
        A(14,15)=X5
C
C
        A(15,9)=(1.,0.)
        A(15,14)=X4
        A(15,15)=X6

C
C
C       ----SET FUNCTION DEFINITION------
C

        IJOB=0

C
C       -------CALL MATRIX INVERSION IMSL ROUTINE-------
C

        CALL LEQ2C(A,N,IA,B,M,IB,IJOB,WA,WK,IER)


        RETURN
        END



CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SUBROUITNE COEFF
C
C               THIS ROUTINE CALCULATED THE COEFFICENTS OF THE
C       UNKNOWN VARIABLES BASED ON THE LENGTH OF THE TRANSMISSION
C       LINE BETWEEN THE ARRAY ELEMENT AND THE PI NETWORK AND THE
C       MAGNITUDE OF THE NETWORK PARAMETRS
C
C       INPUT:  C46,C02,PII,XL13,XC13,XC31
C       OUTPUT: X1,X2,X3,X4,X5,X6
C
C


        SUBROUTINE COEFF(C46,C02,PII,XL13,XC13,XC31,X1,X2,X3,X4,X5,X6)

C
C       ----------DECLARE COMPLEX VARIABLES--------------
C
```

```
      COMPLEX T1,T2,X1,X2,X3,X4,X5,X6,C46,C02,PII,XL13,XC13,XC31
C
C     ----------COMPLEX ALGEBRA CALCULATIONS-------
C

      T1=(PII+(C46/XL13))
      T2=(C02+(PII/XL13))
      X1=-(T1+(C46*XC13))
      X2=(C46/XL13)
      X3=-(T2+(PII*XC13))
      X4=(PII/XL13)
      X5=-(T1+(C46*XC31))
      X6=-(T2+(PII*XC31))


      RETURN
      END
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       SUBROUTINE PATT3
c
c       This subroutine calculates the radiation pattern
c       for the three element equilateral triangular array used
c       to demonstated the design methods of this thesis.
c       The amplitudes and phases of the elements of the array
c       are input from one of the design procedures.  This routine
c       is called by one of the design procedures to display the
c       pattern generated by the solved current distributions.
c
c       INPUT : SCALE,AMP,PH,PI,LENGTH,NUM1,MIT,PHIT
c
c       OUTPUT: XY
c
c

        subroutine patt3(scale,amp,ph,pi,length,num1,Mit,phit)

        real amp(3),p(3),ph(3),x(3),y(3),xy(360,2),Etot(360)
        real scale,d,length,kl,pi,Emax,mit,phit,er(3),ei(3)
        integer num1

c
c       These are the coordinates of the element on the XY plane
c       The length variable is the reciprocal of the separation
c       distance between the elements.  KL is used to convert
c       the geometric phase shift to radians.
c

        kl=2.0*pi/length
        x(1)=0.0
        x(2)=0.5
        x(3)=1.0
        y(1)=0.0
        y(2)=.866025404
        y(3)=0.0

c
c       Use to plot the pattern of a single dipole.
c

        if(num1.eq.1)then
                n=1
                amp(1)=Mit
                p(1)=phit
                x(1)=0.0
                y(1)=0.0
        else
                n=3
        endif


        Emax=0.0

        do I=1,360

                Theta=float(I)*pi/180.0
```

```
          do J=1,n
                  if(x(J).eq.0.0 .and. y(J).eq.0.0)goto 14
C
C           P is the geometric phase shift associated with the physical
C           location of the elements with respect to each other.
C
                  P(J)=kl*sqrt(X(J)**2.+Y(J)**2.)*cos(atan2(Y(J),X(J))-Theta)
  14              p(1)=0.0
C
C           Er(J) is the real component of the complex current on
C           the element and Ei(J) is the imaginary.
C
                  er(J)=amp(J)*cos(ph(J)+P(J))
                  ei(J)=amp(J)*sin(ph(J)+P(J))

          end do
C
C         This is for the single dipole calculations.
C
                  if(numl.gt.1)then
                          do J=2,3
                                  er(1)=er(1)+er(J)
                                  ei(1)=ei(1)+ei(J)
                          end do
                  endif
C
C         Etot(I) is the magnitude of the radiating current
C         as seen at all I angles of observation.
C
                  Etot(I)=Sqrt(er(1)**2.+ei(1)**2.)
                  if(Etot(I).gt.Emax)Emax=Etot(I)

          end do

          write(7,*)' Emax=',Emax
C
C         This 360 degrees loop calculates the XY coordinates of the
C         radiation pattern of the array.
C

          do I=1,360
                  theta=float(I)*pi/180.0
                  ETOT(I)=(Etot(I)*41.0)
                  xy(I,1)=ETOT(I)*cos(theta)
                  xy(I,2)=ETOT(I)*sin(theta)
          end do
C
C         Calls the plotting routine which display the pattern to the screeen.
```

```
c
      call plot(xy,numl)

      return
      end
```

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       This subroutine uses the Plot10 graphics library
c       on the Vax 11/780 at the University of Louisville.
c       It is used to plot radiation patterns.  A matrix
c       containing the X an. Y coordinates of the pattern
c       is passed into this routine and move and draw commands
c       are used to draw this pattern.
c
c       Input : XY,NUM1
c
c

        subroutine plot(xy,num1)

        dimension xy(360,2)
        real radd,ang,x,y,pi
        integer num1


        call grstrt(4014,1)

        if(num1.eq.1)then
                call newpag
        endif

        call window(-750.0,750.0,-750.0,750.0)

        call move(-750.0,0.0)
        call draw(750.0,0.0)
        call move(0.0,750.0)
        call draw(0.0,-750.0)

        call move(xy(1,1),xy(1,2))

        do I=2,360
                x=xy(I,1)
                y=xy(I',2)
                call draw(x,y)
        end do

        call grstop

        return
        end
```

# VITA

PII Redacted

The author [redacted] [redacted] [redacted] [redacted], he graduated from Bardstown High School in Bardstown, Kentucky. He received a Bachelor of Science Degree from the University of Louisville in May of 1985. In May of 1986, he was commissioned an officer in the United States Air Force. The author and his wife Julie now live in Dayton, Ohio where he is currently assigned to the Foreign Technology Division at Wright-Patterson Air Force Base.