Array Failure Correction Using Different Optimization Techniques

Project report submitted in partial fulfillment

of the requirements of the degree of

M.Tech Dual Degree

in

Electrical Engineering
(*Specialization*: *Electronics System and Communication*)

by

Sachin Sagoo

(Roll Number: 711EE1057)

based on research carried out

under the supervision of

Prof. K. Ratna Subhashini

July, 2016

Department of Electrical Engineering
National Institute of Technology Rourkela

Abstract

An approach to synthesize array antenna is proposed on the context of detecting single and multiple fault in any array of elements, analysis of radiation pattern degradation because of the fault, and finding a recovery solution of the array, the recovered excitations fed to the array of healthy elements to get the radiation pattern close to the original form , For this a self-recoverable antenna array is created (SRA). The SRA concept and realization of the different recovery algorithms challenges are discussed. The resulting radiation pattern generated from the new excitation is found by three optimizing algorithm and compared.

***Keywords*: *Self-Recoverable Array(SRA)*; *radiation pattern*; *linear array*; *genetic algorithm*; *differential evolution*; *particle swarm optimization*; *excitations*.**

Contents

Chapter 1

Introduction

1.1 Overview

Due to a constantly increasing number of wireless standards, services, and subscribers who want to enjoy it without disruption of services, preferably at any location, there has been a constant need to offer more robust techniques and technologies that will cope with this demand. In this, we discussed how these techniques will help to find the recovery solution in the case of array element failure. It is desirable that the system has an ability to heal itself as much and fast as possible, before the service crew arrives, which is especially of interest for spaced-based systems or time-critical operations. It is known from antenna array theory that a radiation pattern depends on the excitation magnitude and phase and the locations of the antenna array elements. Due to arbitrariness of the array layout (especially in the case of a random element failure), it is a challenging problem to tackle, even when numerical approaches are utilized.

1.2 Objective

The objective of my project is find a solution to recover the radiation pattern of a faulty antenna array using different optimization techniques.

Chapter 2

Antenna Array Theory

2.1 Introduction

Over the past few decades, since the concept of using antenna arrays instead of a single element has been developed, the performance of a single-element antenna is somewhat limited, researchers have taken on the challenge of providing various array designs to tailor radiation characteristics according to system requirements. Synthesizing an array depends on several factors, such as the requirements of the radiation pattern, the directivity pattern, etc. The radiation pattern depends on the type and number of elements used, and the physical and electrical structure of the array. Numerous variations of antenna structures, as well as the type of elements are available, but for simplicity only one kind of element is used in the whole array structure. In other words, an antenna array is composed of radiating elements in an electrical or geometrical configuration . In an antenna array the total field is calculated by vector addition of each individual element fields radiation. Five parameters that can be use to control an antenna array to shape the pattern properly: the geometrical configuration of the overall array, elements spacing, individual elements excitation amplitude and, excitation phase, and the particular pattern of the individual elements. Many communication applications require a highly directional antenna. Array antennas have higher gain and directivity than an individual radiating element. A linear array consists of elements placed with a uniform spacing in a straight line. The goal of synthesis of antenna array geometry is to determine the physical layout of the array which produces a radiation pattern that is closest to the desired pattern. For the synthesis of the radiation pattern of antenna arrays, various analytical and numerical methods of optimization (End-Fire, Broadside, Hansen-Woodyard, binomial, Dolph-Chebyshev, Neural, Genetic, etc.) were developed and applied. Here, our focus is related to the various analytical methods. In particular, the non-uniform Dolph-Chebyshev and binomial methods will be applied to the synthesis of linear antenna arrays.

2.2 Uniform Spacing, Non-Uniform N-Element Linear Array

In this section, broadside arrays with uniform spacing but non uniform amplitude distribution will be considered (Dolph-Tschebyscheff broadside arrays).

It has been shown analytically that for a given side lobe level the Dolph-Tschebyscheff array produces the smallest beamwidth between the first nulls. Conversely, for a given beamwidth between the first nulls, the Dolph−Tschebyscheff design leads to the smallest possible side lobe level.

- Uniform arrays usually possess the largest directivity. However, super directive antennas possess directivities higher than those of a uniform array.
- Although a certain amount of super directivity is practically possible, super directive array usually require very large currents with opposite phases between adjacent elements. Thus the net current and efficiency of each array are small compared to the corresponding value of an individual element.

2.2.1 Array Factor

An array of element number of isotropic elements 2M is positioned symmetrically along the z-axis, as shown in figure 2.1.

The separation between the element d, and M element are placed on each side of the origin.

the array factor for a broadside array with non uniform amplitude is given by

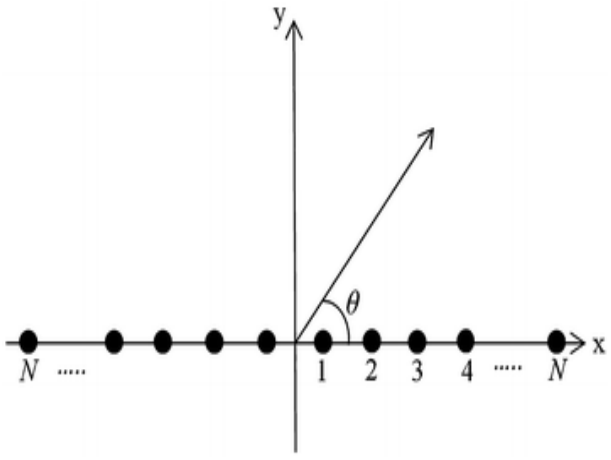

Figure 2.1: A N-ELement Linear array antenna

$$(AF)_{2M} = 2\sum_{n=1}^{M} a_n cos[\frac{(2n-1)}{2} kdcos\theta] \tag{2.1}$$

normalized form

$$(AF)_{2M} = \sum_{n=1}^{M} a_n cos[\frac{(2n-1)}{2} kdcos\theta] \tag{2.2}$$

Where a_n's are the excitation coefficients of the array elements.
array factor for odd(2M+1) number of elements is given by

$$(AF)_{2M+1} = 2\sum_{n=1}^{M} a_n cos[(n-1)kdcos\theta] \tag{2.3}$$

$$(AF)_{2M+1} = 2\sum_{n=1}^{M+1} a_n cos[(n-1)kdcos\theta] \tag{2.4}$$

The amplitude excitation of the centre element is $2a_1$. Equation (2.1) and (2.2) in normalized form

$$(AF)_{2M}(even) = \sum_{n=1}^{M} a_n cos[(2n-1)u] \tag{2.5}$$

$$(AF)_{2M+1}(odd) = 2\sum_{n=1}^{M+1} a_n cos[2(n-1)u] \tag{2.6}$$

$$u = \frac{\pi d}{\lambda} cos\theta \tag{2.7}$$

2.3 Dolph-Tschebyscheff Array

The technique was initially presented by Dolph and researched a while later by others. It is a balance between the binomial and uniform arrays. Its excitation coefficients taken from Tschebyscheff polynomials. A Dolph-Tschebyscheff array with no side lobes can be called as binomial design.

2.3.1 Array Factor

Referring to (2.5) and (2.6), the array factor of odd or even number of elements is given by

$$(AF)_{2M}(even) = \sum_{n=1}^{M} a_n cos[(2n-1)u] \tag{2.8}$$

$$(AF)_{2M+1}(odd) = 2\sum_{n=1}^{M+1} a_n cos[2(n-1)u] \tag{2.9}$$

The largest harmonic of the cosine terms is one less than the total number of elements of the array. Each cosine term, whose argument is an integer times a fundamental frequency, can be rewritten as a series of cosine functions.

$m = 0, \qquad cos(mu) = 1$

$m = 1, \qquad cos(mu) = cos(u)$

$m = 2, \qquad cos(mu) = cos(2u) = 2cos^2u - 1$

$m = 3, \qquad cos(mu) = cos(3u) = 4cos^3u - 3cosu$

$m = 4, \qquad cos(mu) = cos(4u) = 8cos^4u - 8cos^2 + 1$

$m = 5, \qquad cos(mu) = cos(5u) = 16cos^5u - 20cos^3 + 5cosu$

$m = 6, \qquad cos(mu) = cos(6u) = 32cos^6u - 48cos^4 + 18cos^2u - 1$

$m = 7, \qquad cos(mu) = cos(7u) = 64cos^7u - 112cos^5 + 56cos^3u - 7cosu$

$m = 8, \qquad cos(mu) = cos(8u) = 128cos^8u - 256cos^6 + 160cos^4u - 32cos^2u + 1$

$m = 9, \qquad cos(mu) = cos(9u) = 256cos^9u - 576cos^7 + 432cos^5u - 120cos^3u + 9cosu$

if we let

$$z = cosu \tag{2.10}$$

$m = 0, \qquad cos(mu) = 1 = T_0(z)$

$m = 1, \qquad cos(mu) = z = T_1(z)$

$m = 2, \qquad cos(mu) = 2z^2 - 1 = T_2(z)$

$m = 3, \qquad cos(mu) = 4z^3 - 3z = T_3(z)$

$m = 4, \qquad cos(mu) = 8z^4 - 8z^2 + 1 = T_4(z)$

$m = 5, \qquad cos(mu) = 16z^5 - 20z^3 + 5z = T_5(z)$

$m = 6, \qquad cos(mu) = 32z^6 - 48z^4 + 18z^2 - 1 = T_6(z)$

$m = 7, \qquad cos(mu) = 64z^7 - 112z^5 + 56z^3 - 7z = T_7(z)$

$m = 8, \qquad cos(mu) = 128z^8 - 256z^6 + 160z^4 - 32z^2 + 1 = T_8(z)$

$m = 9, \qquad cos(mu) = 256z^9 - 576z^7 + 432z^5 - 120z^3 + 9z = T_9(z)$

and each is related to a Tschebyscheff (Chebyshev) polynomial T_m(z).

Since the array factor of an even or odd number of elements is a summation of cosine terms whose structure is the same as the Tschebyscheff polynomials, the unknown coefficients of the array factor can be determined by equating the series representing the cosine terms of the array factor to the appropriate Tschebyscheff polynomial.

Chapter 3

System Concept and Implimentation

3.1 SRA Concept

The flowchart of the principle steps of SRA.

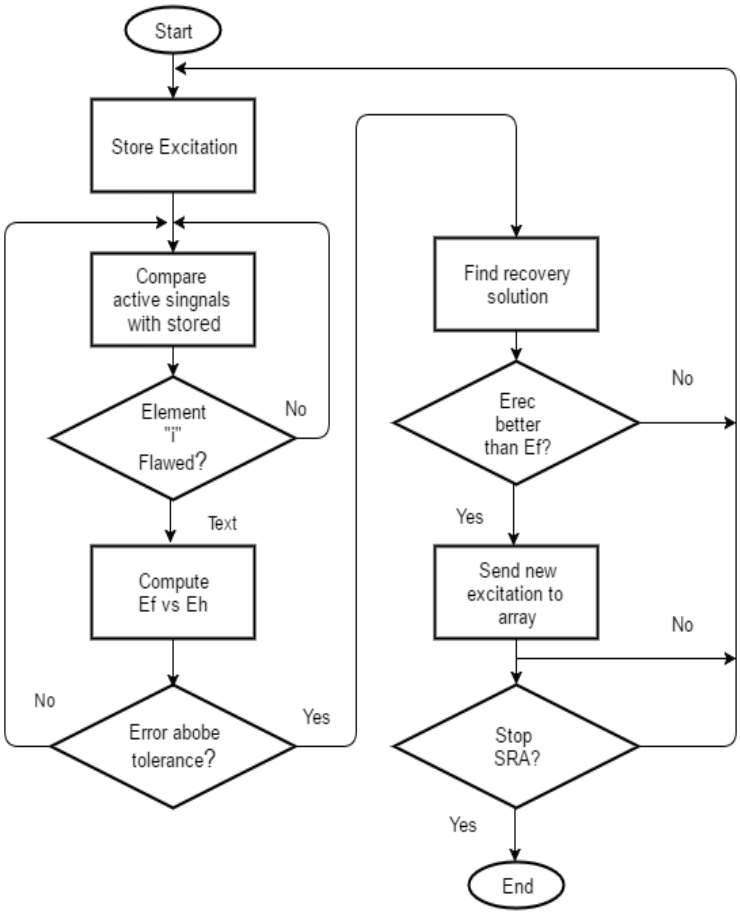

Figure 3.1: The flowchart of the principal steps in the SRA code.

From (2.5)-(2.6), it is known that if any array element fails to deliver power due to some malfunctioning, the radiation pattern of the array will change, possibly severely. For the computation purpose, the array element is considered entirely failed (i.e., its magnitude coefficient equal to zero), even if it actually works at some reduced power. To be able

to analyse the damage due to a given failure, the original set of excitations (magnitudes and phases) is stored. The location of the array elements are considered fixed, which is the case with most arrays in practical use. Thus, when the information about the flawed element is received, the SRA first computes the radiation pattern of the faulty array by (2.1)–(2.2) and compares it to the original radiation pattern. The average cumulative error between the two patterns is defined by

$$e = \sum_{j=\phi_{start}}^{\phi_{end}} w_j.|E_{jo} - E_{jf}| \tag{3.1}$$

If the e is greater than some tolerance level (1.5 dB), the SRA starts exploring for a new set of excitations that will feed the properly working elements of the array and generate the radiation pattern that will be as close as possible to the original radiation pattern. If e < tol, the remaining original excitations are maintained.

3.2 Genetic Algorithm

In my Main Project thesis work the genetic algorithm has been used for the optmization of the antenna excitation to provide the self-recovery solutions. Results are displayed in the next chapter.

3.3 Differential Evolution

In evolutionary computation, differential evolution (DE) is a method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. Such methods are commonly known as metaheuristics as they make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, metaheuristics such as DE do not guarantee an optimal solution is ever found.

DE is used for multidimensional real-valued functions but does not use the gradient of the problem being optimized, which means DE does not require for the optimization problem to be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods. DE can therefore also be used on optimization problems that are not even continuous, are noisy, change over time, etc. [9]

DE optimizes a problem by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones according to its simple formulae, and then keeping whichever candidate solution has the best score or fitness on the optimization problem at hand. In this way the optimization problem is treated as a black box that merely provides a measure of quality given a candidate solution and the gradient is therefore not

needed.

3.3.1 Algorithm

A basic variant of the DE algorithm works by having a population of candidate solutions (called agents). These agents are moved around in the search-space by using simple mathematical formulae to combine the positions of existing agents from the population. If the new position of an agent is an improvement it is accepted and forms part of the population, otherwise the new position is simply discarded. The process is repeated and by doing so it is hoped, but not guaranteed, that a satisfactory solution will eventually be discovered.

Formally, let $f : R^n \rightarrow R$ be the cost function which must be minimized or fitness function which must be maximized. The function takes a candidate solution as argument in the form of a vector of real numbers and produces a real number as output which indicates the fitness of the given candidate solution. The gradient of f is not known. The goal is to find a solution m for which $f(m) \leq f(p)$ for all p in the search-space, which would mean m is the global minimum. Maximization can be performed by considering the function $h := -f$ instead.

Let $x \in R^n$ designate a candidate solution (agent) in the population. The basic DE algorithm can then be described as follows:

- Initialize all agents x with random positions in the search-space.
- Until a termination criterion is met (e.g. number of iterations performed, or adequate fitness reached), repeat the following:

For each agent x in the population do:

- Pick three agents a, b, and c from the population at random, they must be distinct from each other as well as from agent x.
- Pick a random index $R \in \{1, ...n\}$ (n being the dimensionality of the problem to be optimized).
- Compute the agent's potentially new position $y = [y_1....y_n]$ as follows:
  For each i , pick a uniformly distributed number $r_i \equiv U(0, 1)$

  If $r_i <$ CR or $i = R$ then set $y_i = a_i + F \times (b_i - c_i)$ otherwise set $y_i = x_i$ (In essence, the new position is outcome of binary crossover of agent x with intermediate agent

$z = a + F \times b - c$) .)

If $f(y) < f(x)$ then replace the agent in the population with the improved candidate solution, that is, replace x with y in the population.

Pick the agent from the population that has the highest fitness or lowest cost and return it as the best found candidate solution. Note that F $\in [0, 2]$ is called the differential weight and CR $\in$ [0,1] is called the crossover probability, both these parameters are selectable by the practitioner along with the population size NP ≥ 4.

The choice of DE parameters F, CR and NP can have a large impact on optimization performance. Selecting the DE parameters that yield good performance.

3.4 Particle Swarm Optimization

particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position but, is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

PSO is originally attributed to Kennedy, Eberhart and Shi [10] and was first intended for simulating social behaviour, [11] as a stylized representation of the movement of organisms in a bird flock or fish school. The algorithm was simplified and it was observed to be performing optimization.

PSO is a metaheuristic as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, metaheuristics such as PSO do not guarantee an optimal solution is ever found. More specifically, PSO does not use the gradient of the problem being optimized, which means PSO does not require that the optimization problem be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods.

3.4.1 Algorithm

A basic variant of the PSO algorithm works by having a population (called a swarm) of candidate solutions (called particles). These particles are moved around in the search-space according to a few simple formulae. [12] The movements of the particles are guided by their own best known position in the search-space as well as the entire swarm's best known position. When improved positions are being discovered these will then come to guide the movements of the swarm. The process is repeated and by doing so it is hoped, but not guaranteed, that a satisfactory solution will eventually be discovered.

Formally, let $f : R^n \rightarrow R$ be the cost function which must be minimized. The function takes a candidate solution as argument in the form of a vector of real numbers and produces a real number as output which indicates the objective function value of the given candidate solution. The gradient of f is not known. The goal is to find a solution a for which $f(a) \leq f(b)$ for all b in the search-space, which would mean a is the global minimum. Maximization can be performed by considering the function $h = -f$ instead.

Let S be the number of particles in the swarm, each having a position $x_i \in R^n$ in the search-space and a velocity $v_i \in R^n$. Let pi be the best known position of particle i and let g be the best known position of the entire swarm. A basic PSO algorithm is then: [13]

- For each particle $i = 1, ..., S$ do:
- Initialize the particle's position with a uniformly distributed random vector: $x_i \sim U(b_{lo}, b_{up})$, where b_{lo} and b_{up} are the lower and upper boundaries of the search-space.
- Initialize the particle's best known position to its initial position: $p_i \leftarrow x_i$
- If $(f(pi) < f(g))$ update the swarm's best known position:$g \leftarrow p_i$
- Initialize the particle's velocity: $v_i \sim U(-|b_{up} - b_{lo}|, |b_{up} - b_{lo}|)$

Until a termination criterion is met (e.g. number of iterations performed, or a solution with adequate objective function value is found), repeat:

- For each particle $i = 1, ..., S$ do:
- For each dimension $d = 1, ..., n$ do:
- Pick random numbers: $r_p, r_g \sim U(0, 1)$

- Update the particle's velocity: $v_i, d \leftarrow \omega v_i, d + \phi_p r_p(p_i, d - x_i, d) + \phi_g r_g(g_d - x_i, d)$
- Update the particle's position: $x_i \leftarrow x_i + v_i$
- If $(f(xi) < f(pi))$ do:
- Update the particle's best known position: $p_i \leftarrow x_i$
- If $(f(pi) < f(g))$ update the swarm's best known position: $g \leftarrow p_i$
- Now g holds the best found solution.

The parameters ω, ϕ_p, and ϕ_g are selected by the practitioner and control the behaviour and efficacy of the PSO method.

Chapter 4

Simulation and Results

The simulation is done in MATLAB 2015a on PC with Core i5 processor. The recovery solution is generated by optimizing the excitations of the antenna array. The methodology is implemented on uniform spaced Non-uniform(Dolph-Tschebyscheff) linear array antenna to recover the lost radiation pattern. Here the element spacing ($d = \lambda/4$, $d = \lambda/2$) is taken for implementation. For fault analysis one and two element at fault are considered separately and then SRA is implemented on them.

4.1 Genetic Algorithm

The following parameters of genetic algorithm are taken into consideration :
Population size = 70
Generation = 1000
Crossover Probability = 90%
Mutation Rate = 1%
The resulting radiation pattern is plotted as shown in figures.

Element Spacing = λ/4

Recovered results when single element is stimulated as flawed.

Elements = 5

Table 4.1: Recovered excitations for 5-element Linear array antenna with $d = \lambda/4$ when fault occur in single element

Elements	**1**	**2**	**3**	**4**	**5**
A_{Orig}	1	2.4123	3.1396	2.4123	1
A_{recov}	2.391	0	4.682	1.733	1.164

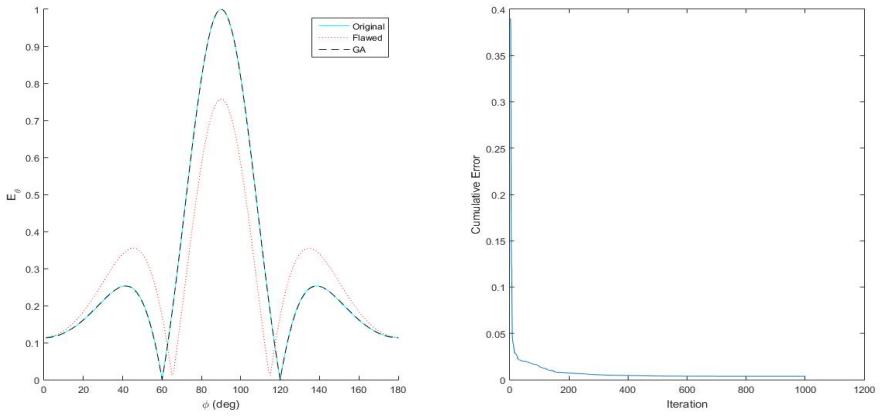

Figure 4.1: A recovery solution for 5-element linear array antenna

Element Spacing = λ/2

Table 4.2: Recovered excitations for 5-element Linear array antenna with $d = \lambda/2$ when fault occur in single element

Elements	**1**	**2**	**3**	**4**	**5**
A_{Orig}	1	2.4123	3.1396	2.4123	1
A_{recov}	0.203	0.031	0	0.250	0.388

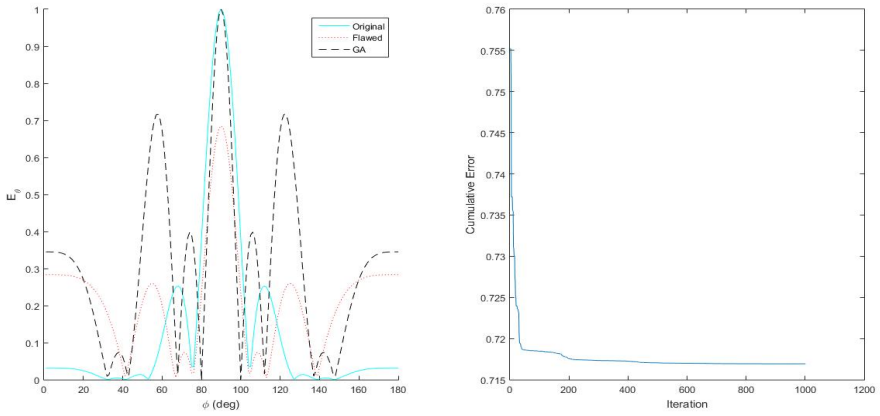

Figure 4.2: A recovery solution for 5-element linear array antenna

4.2 Differential Evolution

The following parameters of Differential Evolution are taken into consideration :

Population = 50

Generation = 1000

The resulting radiation pattern is plotted as shown in figures.

Element Spacing = λ/4

Recovered results when single element is stimulated as flawed.

Elements = 5

Table 4.3: Recovered excitations for 5-element Linear array antenna with $d = \lambda/4$ when fault occur in single element

Elements	**1**	**2**	**3**	**4**	**5**
A_{Orig}	1	2.4123	3.1396	2.4123	1
A_{recov}	2.391	0	4.681	1.735	1.163

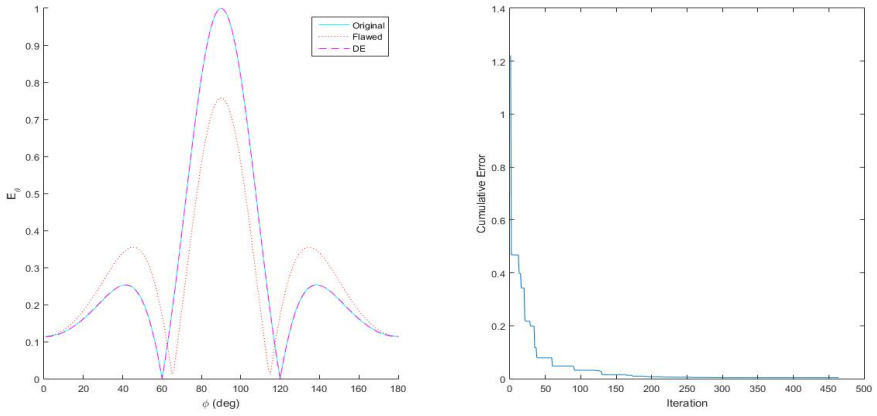

Figure 4.3: A recovery solution for 5-element linear array antenna

Element Spacing = λ/2

Table 4.4: Recovered excitations for 5-element Linear array antenna with $d = \lambda/2$ when fault occur in single element

Elements	**1**	**2**	**3**	**4**	**5**
A_{Orig}	1	2.4123	3.1396	2.4123	1
A_{recov}	0.204	0.032	0	0.250	0.388

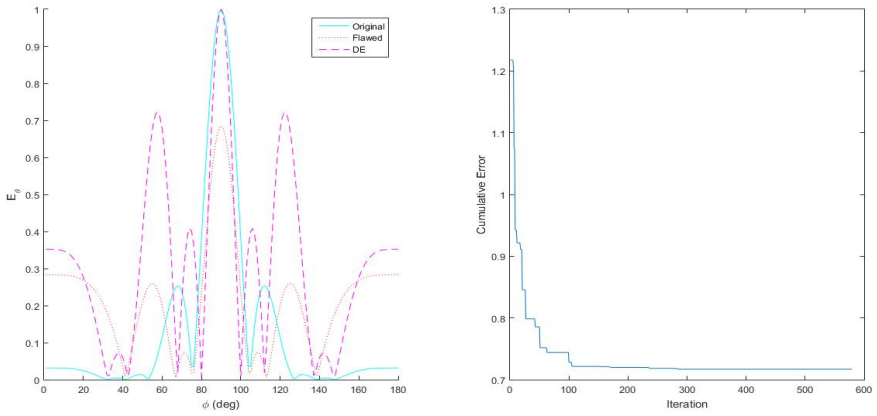

Figure 4.4: A recovery solution for 5-element linear array antenna

4.3 Particle Swarm Optimization

The following parameters of Particle Swarm Optimization are taken into consideration :
Bird in swarm=50
velocity clamping factor=2
cognitive constant=2
social constant=2
Min Inertia weight=0.4
Max Inertia weight=0.9
max iteration=1000

The resulting radiation pattern is plotted as shown in figures.
Element Spacing = λ/4

Recovered results when single element is stimulated as flawed.

Elements = 5

Table 4.5: Recovered excitations for 5-element Linear array antenna with $d = \lambda/4$ when fault occur in single element

Elements	**1**	**2**	**3**	**4**	**5**
A_{Orig}	1	2.4123	3.1396	2.4123	1
A_{recov}	2.392	0	4.680	1.737	1.162

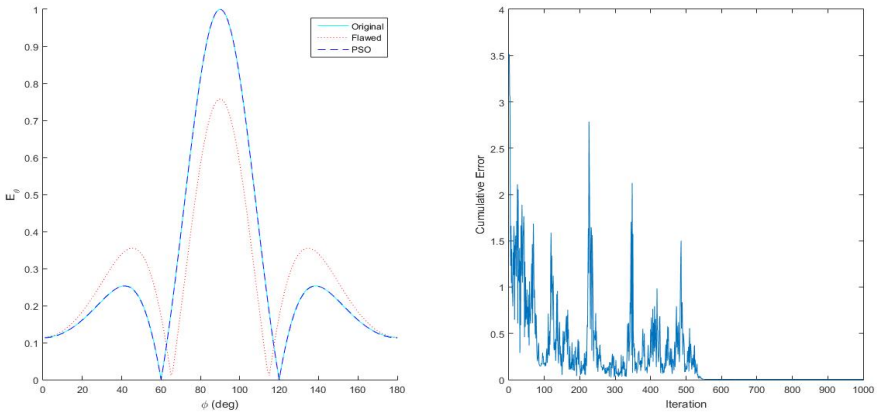

Figure 4.5: A recovery solution for 5-element linear array antenna

Element Spacing = λ/2

Table 4.6: Recovered excitations for 5-element Linear array antenna with $d = \lambda/2$ when fault occur in single element

Elements	**1**	**2**	**3**	**4**	**5**
A_{Orig}	1	2.4123	3.1396	2.4123	1
A_{recov}	0	0	0	0	0.315

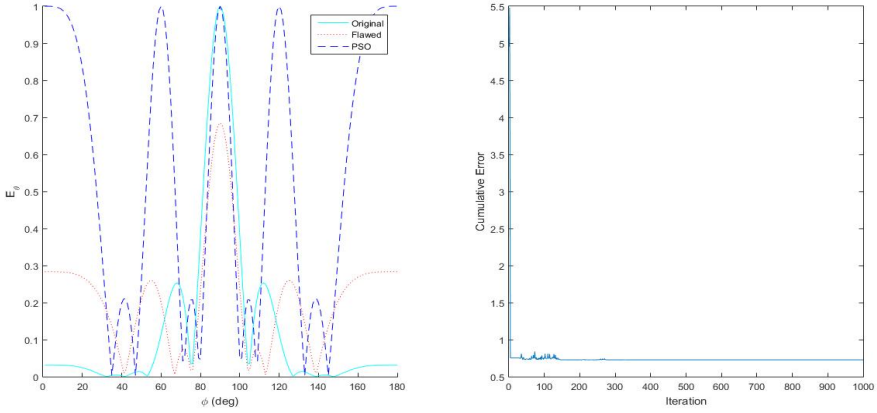

Figure 4.6: A recovery solution for 5-element linear array antenna

4.4 Observations

SRA Recovery Analysis

From all the tests so far, it was noticed that besides the the antenna type, the number of flawed elements and their locations in the array, the fitness of self-recovery solutions also depends on the values of the Algorithm(GA,DE and PSO) parameters and the type of the array excitation.

A classic Dolph–Chebyshev linear array design with an SLL of -30 dB is used as a reference.

After optimization of the antenna array by GA, DE and PSO techniques the following cumulative errors are recorded.

Table 4.7: Cumulative Error found after optimization by GA,DE and PSO when element spacing $d = \lambda/4$

Optimization Techniques	**No. of Elements**	**No of Faulty Elements**	**Cumulative Error**
GA	5	1	0.0141800
DE	5	1	0.0319818
PSO	5	1	0.0482042

A. 5 Element array at $d = \lambda/4$, 1 element failure correction.
Fig. 4.7(b) depicts the fitness progress curves, Notice that convergence is observed for all the above cases before 1000 generations. The cumulative error after 600 generations is the lowest.

From the Table 4.7 shows the fitness value of the technique used for optimization. From the table we concluded that all three technique have provided nearly same fitness value. convergence graph show that GA and DE technique have fast convergence rate as compared to PSO. The new recovery solution is plotted in fig 4.7(a)

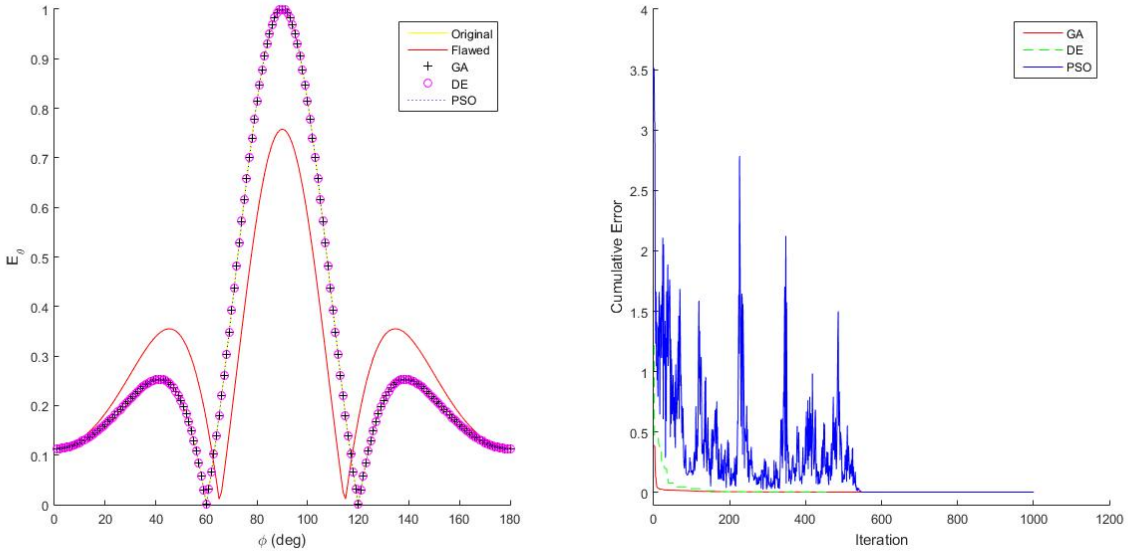

Figure 4.7: Comparison between the recovery solution for 5-element found by GA,DE and PSO at $d = \lambda/4$

B. 5 Element array at $d = \lambda/2$, 1 element failure correction.

Table 4.8: Cumulative Error found after optimization by GA,DE and PSO at $d = \lambda/2$

Optimization Techniques	**No. of Elements**	**No of Faulty Elements**	**Cumulative Error**
GA	5	1	0.7169122
DE	5	1	0.7169144
PSO	5	1	0.7467602

Fig. 4.8(b) depicts the fitness progress curves, Notice that convergence is observed for all the above cases before 1000 generations. The cumulative error for GA and DE after 200 generations is the lowest. But PSO conversed on to the local optimal solution.

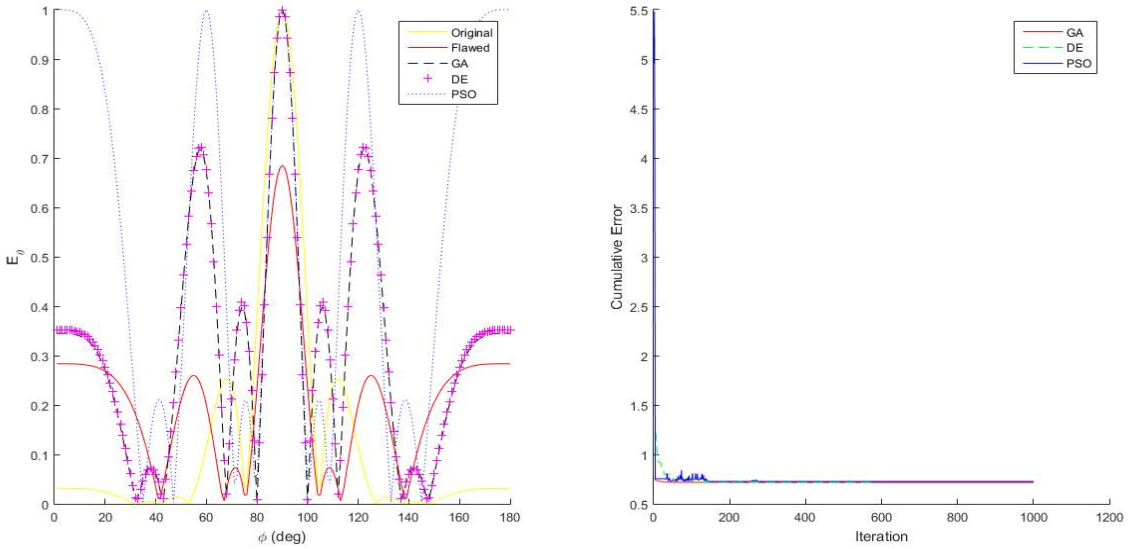

Figure 4.8: Comparison between the recovery solution for 5-element found by GA,DE and PSO at $d = \lambda/2$

From the Table 4.8 shows the fitness value of the technique used for optimization. From the table we concluded that GA and DE have provided better fitness value than PSO. Convergence graph show that GA and DE technique have fast convergence as compared to PSO and PSO fail to find the recovery solution as the distance between increases. The new recovery solution is plotted in fig 4.8(a).

Chapter 5

Conclusion

5.1 Conclusion

i. A recovery solution for different number of elements has been obtained by optimizing the excitation of linear array antenna by three different algorithm.

ii. GA and DE have proven usefull for finding the better recovery solution.

iii. SRA is bounded by element spacing d.

iV. PSO fails to find the recovery solution as the element spacing increases above $\lambda/2$.

References

[1] T. J. Peters, "A conjugate gradient-based algorithm to minimize the sidelobe level of planar arrays with element failures," *Antennas and Propagation, IEEE Transactions on*, vol. 39, no. 10, pp. 1497–1504, 1991.

[2] R. J. Mailloux, "Array failure correction with a digitally beamformed array," *Antennas and Propagation, IEEE Transactions on*, vol. 44, no. 12, pp. 1543–1550, 1996.

[3] B.-K. Yeo and Y. Lu, "Array failure correction with a genetic algorithm," *Antennas and Propagation, IEEE Transactions on*, vol. 47, no. 5, pp. 823–828, 1999.

[4] D. Marcano and F. Durán, "Synthesis of antenna arrays using genetic algorithms," *Antennas and Propagation Magazine, IEEE*, vol. 42, no. 3, pp. 12–20, 2000.

[5] J. Rodriguez, F. Ares, H. Palacios, and J. Vassal'lo, "Finding defective elements in planar arrays using genetic algorithms," *Progress In Electromagnetics Research*, vol. 29, pp. 25–37, 2000.

[6] S. Nakazawa, S. Tanaka, and T. Murata, "Evaluation of degradation of shaped radiation pattern caused by excitation coefficient error for onboard array-fed reflector antenna," in *Antennas and Propagation Society International Symposium, 2004. IEEE*, vol. 3. IEEE, 2004, pp. 3047–3050.

[7] M. Joler, "How fpgas can help create self-recoverable antenna arrays," *International Journal of Antennas and Propagation*, vol. 2012, 2012.

[8] R. L. Haupt and S. E. Haupt, *Practical genetic algorithms*. John Wiley & Sons, 2004.

[9] P. Rocca, G. Oliveri, and A. Massa, "Differential evolution as applied to electromagnetics," *IEEE Antennas and Propagation Magazine*, vol. 53, no. 1, pp. 38–49, 2011.

[10] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. IEEE, 1998, pp. 69–73.

[11] J. Kennedy, "The particle swarm: social adaptation of knowledge," in *Evolutionary Computation, 1997., IEEE International Conference on*. IEEE, 1997, pp. 303–308.

[12] Y. Zhang, S. Wang, and G. Ji, "A comprehensive survey on particle swarm optimization algorithm and its applications," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[13] M. Clerc, "Standard particle swarm optimisation," 2012.