Rectangular Microstrip Patch Antenna Design Optimization, using Particle Swarm Optimization Algorithm

INSTITUTE OF ENGINEERING AND MANAGEMENT

May 18, 2013

Authored by: Saptarshi Ghosh, Suman Chakroborty, Sushmita Saha, Sougata Majumdar, Shubhojit Chaterjee

Mentor: *Malay Ganguly*

Acknowledgement

On the first hand we would like to express our highest gratitude and thanks to our mentor Mr. **Malay Ganguly** who had helped us at each and every point of the project work with his dedication, with his comments, suggestions and guidance which put us on the right path to fulfill the requirement without which this situation wouldn't have aroused of presenting this project work.

I also would like to thank my friends for their support and constant help in lending me their laptop with higher configuration in order to run the simulations for this work.

Support from our college- **INSTITUTE OF ENGINEERING AND MANAGEMENT** is much appreciated in providing us the facility which encouraged us to work more and more.

And last but not the least we would like to express our hearty thanks to GOD and all those who supported us directly or indirectly to complete this task.

- *Saptarshi Ghosh*

Contents

Section-A

- ❖ **Introduction**
- ❖ **Overview of the project**

• Introduction

A major role is played by communication engineering in our present day life ensuring instant connectivity from one part of the world with another. Antennas play a very important role in the field of wireless communications. They are the backbone and almost everything in the wireless communication without which the world could have not reached at this age of technology. It is an electrical device which converts electric power into radio waves and vice versa.

There are several types of antenna depending on the need and application .There are dipole antenna, patch antenna, parabolic reflector antenna, slot antenna etc.Micro strip patch antennas play a very significant role in today's world of wireless communication systems . It consists of a flat rectangular sheet or "patch" of metal, mounted over a larger sheet of metal called a ground plane. Patch antennas are simple to fabricate and easy to modify and customize. However two major disadvantages are low gain and narrow bandwidth when high dielectric constant material is used for fabrication of the micro strip antenna.

In this project we discuss analyze and optimize the design of a rectangular micro strip patch antenna using Particle Swarm Optimization Algorithm.

• Overview of the project

This project focuses on the design of rectangular micro strip patch antenna .The design parameters, design method and optimization technique (to obtain the best radiation at the particular operating frequency) are described in detail.

There are several optimization algorithms, such as genetic algorithm, differential optimization technique ant colony optimization etc that can be used for the antenna design optimization .In this project Particle swarm optimization (PSO) algorithm is used to optimize the antenna design.

Matlab R2009 software tool is used for the code implementation of the PSO algorithm for patch antenna design optimization.

The design result thus obtained is simulated using ZELAND IE3D antenna simulation software and the performance of the design is observed and analyzed.

Section-B

- ❖ **Particle Swarm Optimization Algorithm**
- ❖ **Basics of Micro strip Patch Antenna**

• Particle Swarm Optimization Algorithm

Introduction

Particle swarm optimization (PSO) is a population-based approach for solving continuous and discrete optimization problems.
In particle swarm optimization, simple software agents, called *particles*, move in the search space of an optimization problem. The position of a particle represents a candidate solution to the optimization problem at hand. Each particle searches for better positions in the search space by changing its velocity according to rules originally inspired by behavioural models of bird flocking.
Particle swarm optimization belongs to the class of swarm intelligence techniques that are used to solve optimization problems.

History

Particle swarm optimization was introduced by Kennedy and Eberhart (1995). It has roots in the simulation of social behaviours using tools and ideas taken from computer graphics and social psychology research.

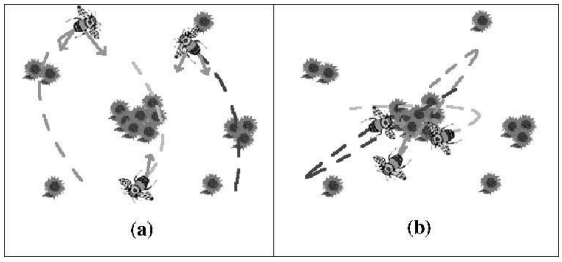
Within the field of computer graphics, the first antecedents of particle swarm optimization can be traced back to the work of Reeves (1983), who proposed particle systems to model objects that are dynamic and cannot be easily represented by polygons or surfaces. Examples of such objects are fire, smoke, water and clouds. In these systems, particles are independent of each other and their movements are governed by a set of rules. Some years later, Reynolds (1987) used a particle system to simulate the collective behaviour of a flock of birds. In a similar kind of simulation, Heppner and Grenander (1990) included a *roost* that was attractive to the simulated birds. Both

models inspired the set of rules that were later used in the original particle swarm optimization algorithm.

Social psychology research, in particular the dynamic theory of social impact (Nowak, Szamrej & Latané, 1990), was another source of inspiration in the development of the first particle swarm optimization algorithm (Kennedy, 2006). The rules that govern the movement of the particles in a problem's search space can also be seen as a model of human social behaviour in which individuals adjust their beliefs and attitudes to conform to those of their peers (Kennedy & Eberhart 1995).

Working:

PSO can best be understood through an analogy similar to the one that led to the development of the PSO.Imagine a swarm of bees in a field. Their goal is to find in the field the location with the highest density of flowers. Without any knowledge of the field a priori, the bees begin in random locations with random velocities looking for flowers. Each bee can remember the locations that it found the most flowers, and somehow knows the locations where the other bees found an abundance of flowers. Torn between returning to the location where it had personally found the most flowers, or exploring the location reported by others to have the most flowers, the ambivalent bee accelerates in both directions altering its trajectory to fly somewhere between the two points depending on whether nostalgia or social influence dominates its decision.



Along the way, a bee might find a place with a higher concentration of flowers than it had found previously. It would then be drawn to this new location as well as the location of the most flowers found by the whole swarm. Occasionally, one bee may fly over a place with more flowers than had been encountered by any bee in the swarm. The whole swarm would then be drawn toward that location in additional to their own personal discovery.

In this way the bees explore the field: overflying locations of greatest concentration, then being pulled back toward them. Constantly, they are checking the territory they fly over against previously encountered locations of highest concentration hoping to find the absolute highest concentration of flowers. Eventually, the bees' flight leads them to the one place in the field with the highest concentration of flowers. Soon, all the bees swarm around this point.

Unable to find any points of higher flower concentration, they are continually drawn back to the highest flower concentration.

PSO Language

The language used to discuss the PSO follows from the analogy of particles in a swarm, much like the analogy presented above. Table I shows some of the key terminology.

Key PSO Vocabulary	
Particle/Agent	One single individual in the swarm.
Location/Position	An agent's N-dimensional coordinates which represents a solution to the problem.
Swarm	The entire collection of agents.
Fitness	A single number representing the goodness of a given solution (represented by a location in solution space)
pbest	The location in parameter space of the best fitness returned for a specific agent.
gbest	The location in parameter space of the best fitness returned for the entire swarm.
V_{max}	The maximum allowed velocity in a given direction.

Some key terms used to describe PSO

A more detailed description of some of these key terms is given below:

1) Particle or Agent: Each individual in the swarm (bees in the analogy above) is referred to as a particle or agent. All the particles in the swarm act individually under the same governing principle: accelerate toward the best personal and best overall location while constantly checking the value of its current location.

2) *Position:* In the analogy above *position* referred to a bee's place in the field. This is represented by coordinates on the - plane. In general, however, we can extend this idea into any -dimensional space according the problem at hand. This -dimensional space is the solution space for the problem being optimized, where any set of coordinates represents a solution to the problem. In the analogy above the solution is a physical location on the - plane, but this could just as easily represent amplitude and phase of element excitation in a phased array. In general these can be any values needed to be optimized. Reducing the optimization problem to a set of values that could represent a position in solution space is an essential step in utilizing the PSO.

3) *Fitness:* As in all evolutionary computation techniques there must be some function or method to evaluate the goodness of a position. The fitness function must take the position in the solution space and return a single number representing the value of that position. In the analogy above the fitness function would simply be the density of

flowers: the higher the density, the better the location. In general this could be antenna gain, weight, peak cross-polarization, or some kind of weighted sum of all these factors. The fitness function provides the interface between the physical problem and the optimization algorithm.

4) *pbest:* In the analogy above each bee remembers the location where it personally encountered the most flowers. This location with the highest fitness value personally discovered by a bee is known as the personal best or *pbest*. Each bee has its own *pbest* determined by the path that it has flown. At each point along its path the bee compares the fitness value of its current location to that of *pbest*. If the current location has a higher fitness value, *pbest* is replaced with its current location.

5) *gbest:* Each bee also had some way of knowing the highest concentration of flowers discovered by the entire swarm. This location of highest fitness encountered is known as the global best or *gbest*. For the entire swarm there is one *gbest* to which each bee is attracted. At each point along their path every bee compares the fitness of their current location to that of *gbest*. If any bee is at a location of higher fitness, *gbest* is replaced by that bee's current position.

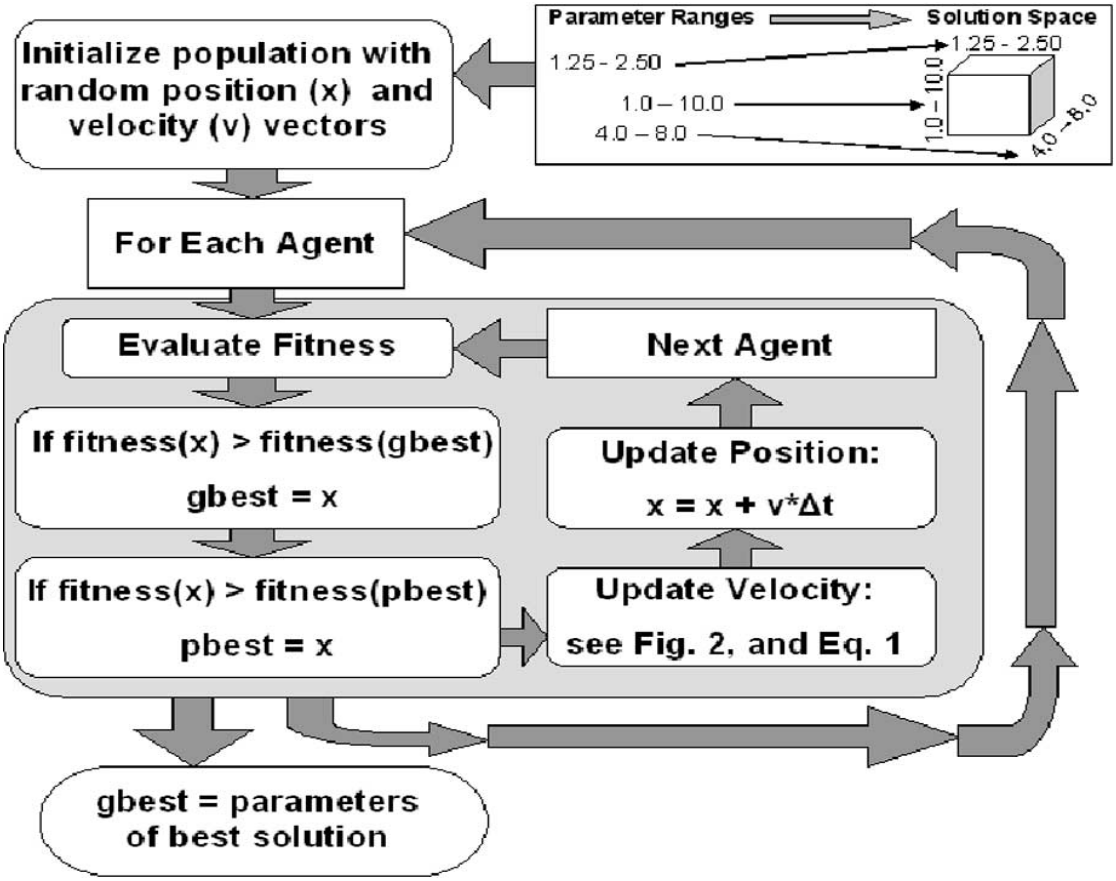
Concept of fitness function, pbest and gbest location

As an analogy to the bee example,

- Position of a bee is represented by x-y co-ordinates. The idea can be extended to N dimension.
- **Fitness function** determines the magnitude of concentration of flower. Determines how well the parameters x-y or further dimension fits. It must return a single number representing the value of that position.
- Bee remembers the location where it personally encountered the most flowers. In its path for a given interval of time, the position that gives the maximum value of the fitness function is the **local best position or pbest location.**
- For the same interval of time, all other bee in the swarm will have their own pbest location. The one having highest fitness value of all will be the **global best position or gbest.**

Development of the PSO algorithm

1) *Defining the Solution Space:* The first step toward implementation of the PSO is to pick the parameters that need to be optimized and give them a reasonable range in which to search for the optimal solution. This requires specification of a minimum and maximum value for each dimension in an N-dimensional optimization. This is referred to as $Xmin_n$ and $Xmax_n$, respectively, where n ranges from 1 to N.



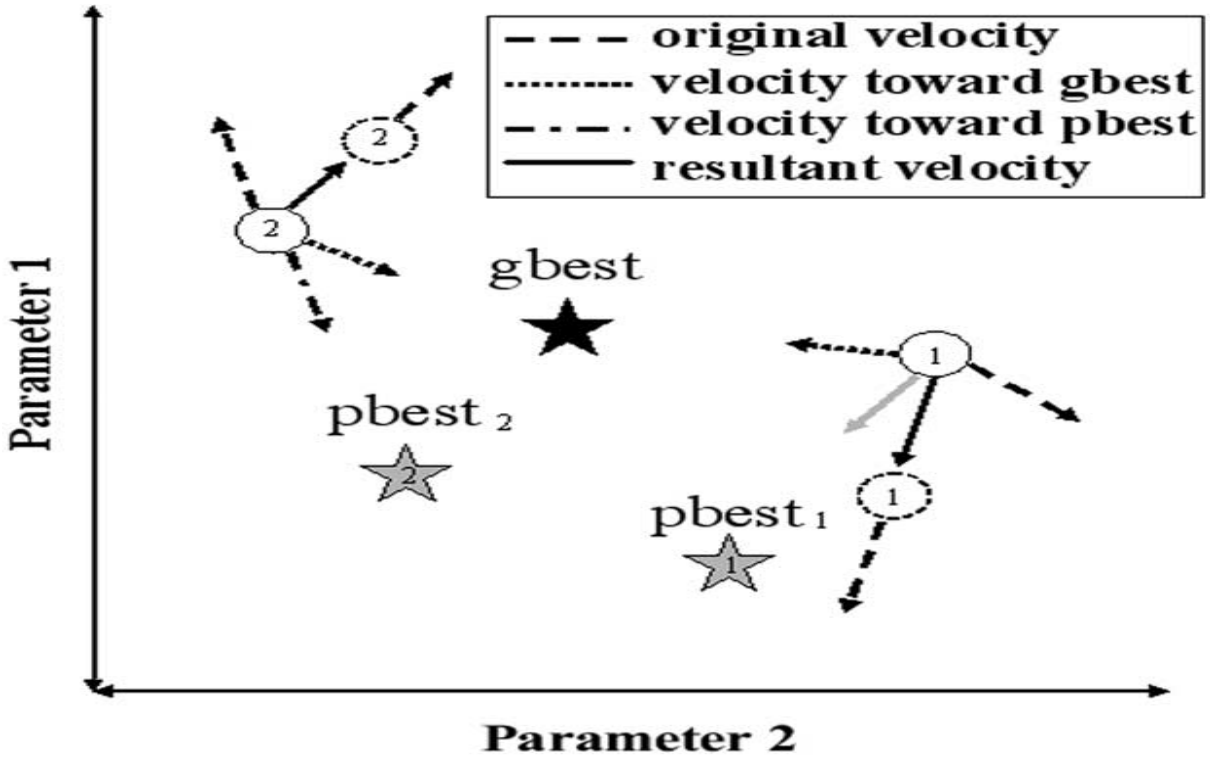
Flow chart depicting the PSO algorithm

2) *Defining a Fitness Function:* This important step provides the link between the optimization algorithm and the physical world. It is critical that a good function be chosen that accurately represents, in a single number, the good- ness of the solution. The fitness function should exhibit a functional dependence that is relative to the importance of each characteristic being optimized. The fitness function and

the solution space must be specifically developed for each optimization; the rest of the implementation, how- ever, is independent of the physical system being optimized.

3) *Initializing Random Swarm Location and Velocities:* To begin searching for the optimal position in the solution space, each particle begins at its own random location with a velocity that is random both in its direction and magnitude. Since its initial position is the only location encountered by each particle at the run's start, this position becomes each particle's respective *pbest*. The first *gbest* is then selected from among these initial positions.

4) *Systematical Fly of the Particles Through the Solution Space:* Each particle must then be moved through the solution space as if it were a bee in a swarm. The algorithm acts on each particle one by one, moving it by a small amount and cycling through the entire swarm. The following steps are enacted on each particle individually



Individual particles (1 and 2) are accelerated toward the location of the best solution *gbest*, and the location of their own personal best *pbest*, in a 2-D parameter space.

a)*Evaluate the Particle's Fitness, Compare to gbest, pbest:* The fitness function, using the coordinates of the particle in solution space, returns a fitness value to be assigned to the current location. If that value is greater than the value at the respective *pbest* for that particle, or the global *gbest,* then the appropriate locations are replaced with the current location.

b) *Updating the Particle's Velocity:* The manipulation of a particle's velocity is the core element of the entire optimization. Careful understanding of the equation used to determine the velocity is the key to understanding the optimization as a whole. The velocity of the particle is changed according to the relative locations of *pbest* and *gbest*. It is accelerated in the directions of these locations of greatest fitness according to the following equation:

$$v_n = w * v_n + c_1 \text{ rand}() * (p_{best,n} - x_n) + c_2 \text{ rand}() * (g_{best,n} - x_n) \quad (1)$$

Where v_n is the velocity of the particle in the nth dimension and is x_n the particle's coordinate in the nth dimension. This calculation is done for each of the N dimensions in an N-dimensional optimization. Apparent from this equation, the new velocity is simply the old velocity scaled by and increased in the direction of gbest and pbest for that particular dimension. Now, c_1 and c_2 are scaling factors that determine the relative "pull" of pbest and gbest. These are sometimes referred to as the cognitive and social rates, respectively; c_1 is a factor determining how much the particle is influenced by the memory of his best location, and c_2 is a factor determining how much the particle is influenced by the rest of the swarm. Increasing c_1 encourages exploration of the solution space as each particle moves toward its own pbest; increasing c_2 encourages exploitation of the supposed global maximum.

The random number function rand() returns a number between 0.0 and 1.0. It is generally the case that the two appearances of the rand() function in (1) represent two separate calls to the function. Most implementations use two independent random numbers to stochastically vary the relative pull of *gbest* and *pbest*. This introduction of a random element into the optimization is intended to simulate the slight unpredictable component of natural swarm behaviour; w is known as the "inertial weight," and this number (chosen to be between 0.0 and 1.0) determines to what extent the particle remains along its original course unaffected by the pull of *gbest* and *pbest*. This too is a way to balance exploration and exploitation. The motion of the particle can be traced based on (1). The particles furthest from *gbest* or *pbest* feel the greatest "pull" from the respective locations, and therefore move toward them more rapidly than a particle that is closer. The particle continues to gain speed in the direction of the locations of greatest fitness until they pass over them. At that point they begin to be pulled back in the opposite direction. It is this "overflying" of the local and global maxima that many believe is one secret to the PSOs success.

c) *Moving of the Particle:* Once the velocity has been determined it is simple to move the particle to its next location. The velocity is applied for a given time-step and new coordinate x_n is computed for each of the N dimensions according the following equation:

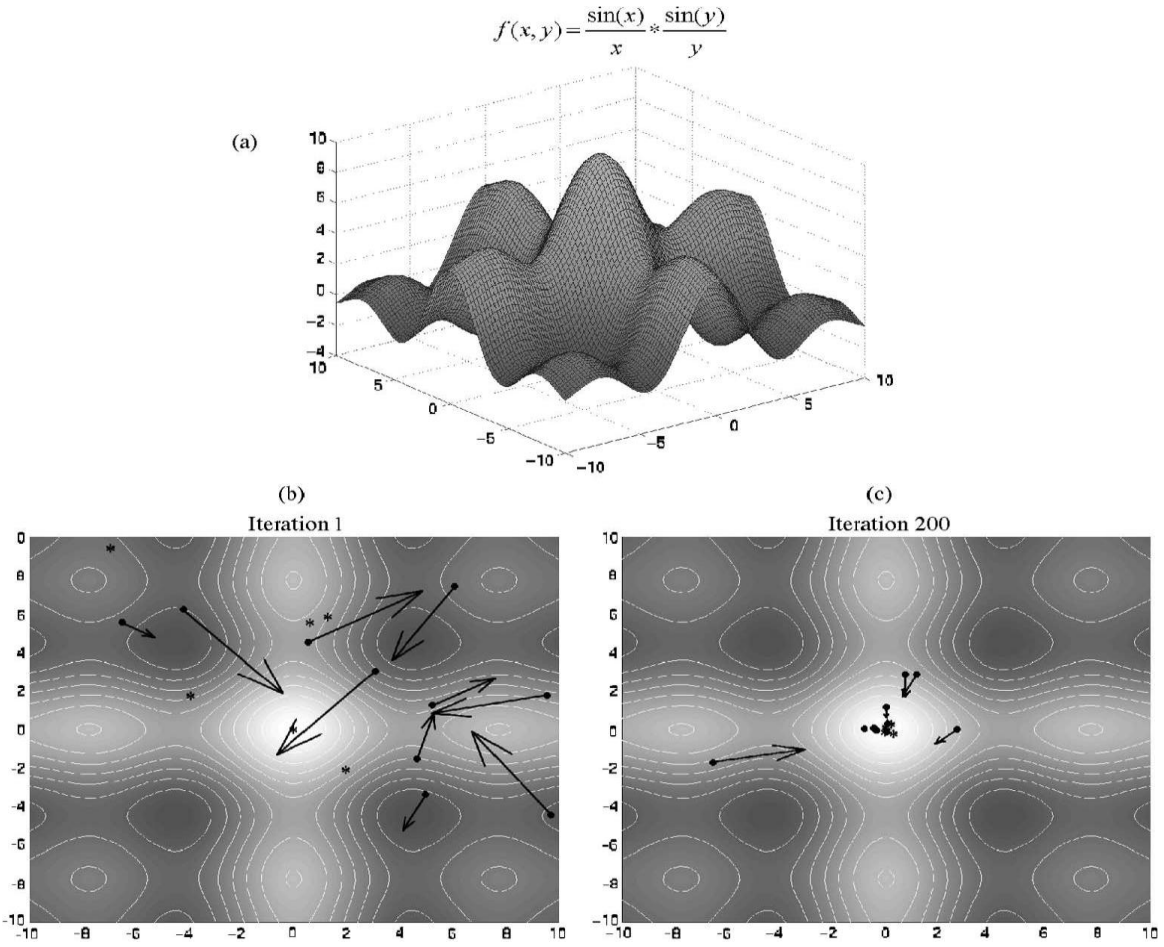
$$x_n = x_n + \Delta t * v_v. \qquad (2)$$

The particle is then moved to the location calculated by (2). The composite nature of this algorithm composed of several independent agents makes it especially conducive to implementation on parallel processors.

5) *Repeat:* After this process is carried out for each particle in the swarm, the process is repeated starting at Step 4). In this way the particles move for discrete time intervals before being evaluated. It is as though a snapshot is taken of the entire swarm every second. At that time the positions of all the particles are evaluated, and corrections are made to the positions of *pbest*, and *gbest* before letting the particles fly around for another second. Repetition of this cycle is continued until the termination criteria are met. There are several methods to determine these termination criteria. The criterion most often used in optimizations with the UCLA-PSO(developed by Kennedy and Eberhart) is a maximum iteration number. With this termination condition, the PSO ends when the process starting with Step 4) has been repeated a user-defined number of times. Another criterion available with the UCLA-PSO is a target fitness termination condition. With this option the PSO is run for the user-defined number of iterations, but at any time if a solution is found that is greater than or equal to the target fitness value, then the PSO is stopped at that point. This is useful when one has a very specific engineering goal for the value of the fitness function, and is not necessarily concerned with finding the "best" solution. In some cases if a solution is found to be better than the target fitness, then the solution is good enough and there is no reason to continue the run. A final criterion employed by the UCLA-PSO is a minimum standard deviation (minSTD) criterion. This condition, which can be used in any combination with the previously mentioned criteria, compares the mean STD of all the particles to the minSTD. If the current STD is less than the user-determined minSTD, then all the particles are said to be satisfactorily converged around the *gbest*, and the run is ended with the assumption that the PSO has stagnated around this point. For this termination criterion it is important to compute the STD relative to the dynamic range of each dimension and consider the effect of the inertial weight.

Example:

The locations of the particles and the arrows show the direction and magnitude of their velocity. The asterisks represent the various locations of the *pbests*. The best of these locations becomes the *gbest*. After the first iteration, there is a wide spread in the location of the particles and *pbests*. The velocity vectors are relatively large, indicating much movement in the early iterations. After the two-hundredth iteration, however, the particles and their *pbests* are localized around the global maximum. The velocities are relatively small due the proximity of the particles to the location of *pbest* and *gbest*. Here the "swarming" of the particles is visible as particles that have flown over the *gbest* are pulled back toward the global maximum.

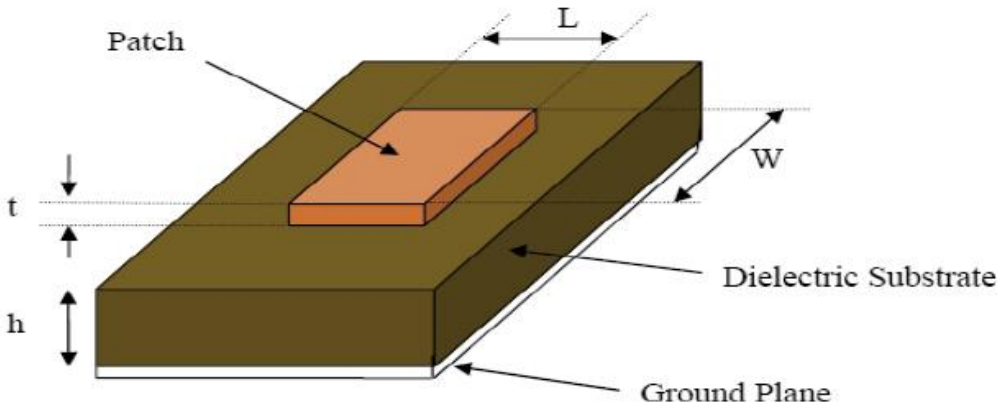


• Basics of Rectangular Microstrip Patch Antenna

Microstrip antennas are attractive due to their light weight, conformability and low cost. These antennas can be integrated with printed strip-line feed networks and active devices. This is a relatively new area of antenna engineering. The radiation properties of micro strip structures have been known since the mid 1950's.The application of this type of antennas started in early 1970's when conformal antennas were required for missiles. Rectangular and circular micro strip resonant patches have been used extensively in a variety of array configurations. A major contributing factor for recent advances of microstrip antennas is the current revolution in electronic circuit miniaturization brought about by developments in large scale integration. As conventional antennas are often bulky and costly part of an electronic system, micro strip antennas based on photolithographic technology are seen as an engineering breakthrough.

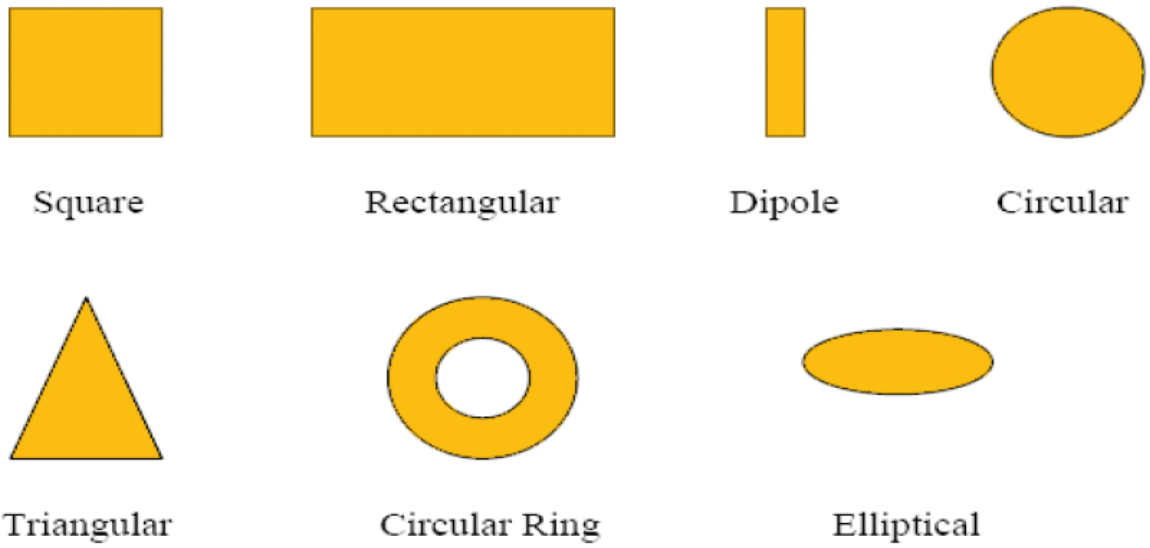
Introduction

In its most fundamental form, a Microstrip Patch antenna consists of a radiating patch on one side of a dielectric substrate which has a ground plane on the other side . The patch is generally made of conducting material such as copper or gold and can take any possible shape. The radiating patch and the feed lines are usually photo etched on the dielectric substrate.



Structure of a Micro strip Patch Antenna

The patch is generally square , rectangular, circular, triangular, and elliptical or some other common shape.For a rectangular patch, the length L of the patch is usually $0.3333\lambda_o < L < 0.5\ \lambda_o$, where λ_o is the free-space wavelength. The patch is selected to be very thin such that $t << \lambda_o$ (where *t* is the patch thickness). The height *h* of the dielectric substrate is usually $0.003\ \lambda_o \leq h \leq 0.05\ \lambda_o$. The dielectric constant of the substrate (ε_r) is typically in the range $2.2 \leq \varepsilon_r \leq 12$.



Common shapes of microstrip patch elements

Microstrip patch antennas radiate primarily because of the fringing fields between the patch edge and the ground plane. For good antenna performance, a thick dielectric substrate having a low dielectric constant is desirable since this provides better efficiency, larger bandwidth and better radiation . However, such a configuration leads to a larger antenna size. In order to design a compact Microstrip patch antenna, substrates with higher dielectric constants must be used which are less efficient and result in narrower bandwidth. Hence a trade-off must be realized between the antenna dimensions and antenna performance.

Advantages and Disadvantages

Microstrip patch antennas are increasing in popularity for use in wireless applications due to their low-profile structure. Therefore they are extremely compatible for embedded antennas in handheld wireless devices such as cellular phones, pagers etc... The telemetry and communication antennas on missiles need to be thin and conformal and are often in the form of Microstrip patch antennas. Another area where they have been used successfully is in Satellite communication. Some of their principal advantages discussed by Kumar and Ray are given below:

- Light weight and low volume.
- Low profile planar configuration which can be easily made conformal to host surface.
- Low fabrication cost, hence can be manufactured in large quantities.
- Supports both, linear as well as circular polarization.
- Can be easily integrated with microwave integrated circuits (MICs).
- Capable of dual and triple frequency operations.
- Mechanically robust when mounted on rigid surfaces.

Microstrip patch antennas suffer from more drawbacks as compared to conventional antennas. Some of their major disadvantages discussed by [9] and Garg et al [10] are given below:

- Narrow bandwidth
- Low efficiency
- Low Gain
- Extraneous radiation from feeds and junctions
- Poor end fire radiator except tapered slot antennas
- Low power handling capacity.
- Surface wave excitation

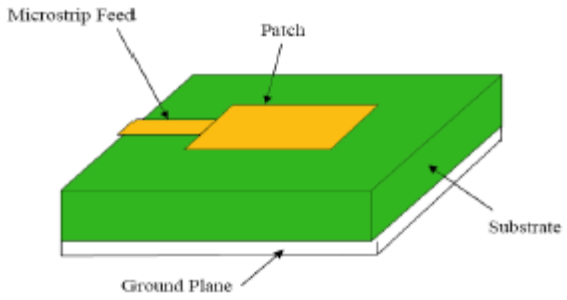
Microstrip patch antennas have a very high antenna quality factor (Q). It represents the losses associated with the antenna where a large Q leads to narrow bandwidth and low efficiency. Q can be reduced by increasing the thickness of the dielectric substrate. But as the thickness increases, an increasing fraction of the total power delivered by the source goes into a surface wave. This surface wave contribution can be counted as an unwanted power loss since it is ultimately scattered at the dielectric bends and causes degradation of the antenna characteristics. Other problems such as lower gain and lower power handling capacity can be overcome by using an array configuration for the elements.

Feed Techniques

Microstrip patch antennas can be fed by a variety of methods. These methods can be classified into two categories- contacting and non-contacting. In the contacting method, the RF power is fed directly to the radiating patch using a connecting element such as a microstrip line. In the non-contacting scheme, electromagnetic field coupling is done to transfer power between the microstrip line and the radiating patch . The four most popular feed techniques used are the microstrip line, coaxial probe (both contacting schemes), aperture coupling and proximity coupling (both non-contacting schemes).

Microstrip Line Feed

In this type of feed technique, a conducting strip is connected directly to the edge of the Microstrip patch. The conducting strip is smaller in width as compared to the patch and this kind of feed arrangement has the advantage that the feed can be etched on the same substrate to provide a planar structure.

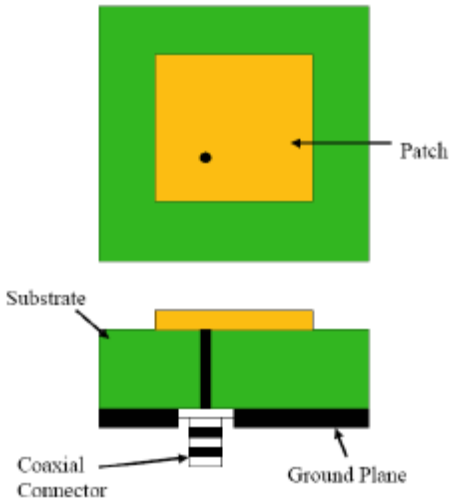


Microstrip Line Feed

The purpose of the inset cut in the patch is to match the impedance of the feed line to the patch without the need for any additional matching element. This is achieved by properly controlling the inset position. Hence this is an easy feeding scheme, since it provides ease of fabrication and simplicity in modeling as well as impedance matching. However as the thickness of the dielectric substrate being used, increases, surface waves and spurious feed radiation also increases, which hampers the bandwidth of the antenna [5]. The feed radiation also leads to undesired cross polarized radiation.

Coaxial Feed

The Coaxial feed or probe feed is a very common technique used for feeding Microstrip patch antennas. As seen from Figure 2.4, the inner conductor of the coaxial connector extends through the dielectric and is soldered to the radiating patch, while the outer conductor is connected to the ground plane.



Probe fed Rectangular Microstrip Patch Antenna

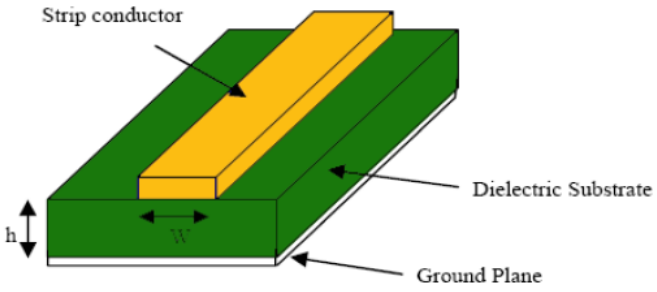
The main advantage of this type of feeding scheme is that the feed can be placed at any desired location inside the patch in order to match with its input impedance. This feed method is easy to fabricate and has low spurious radiation. However, a major disadvantage is that it provides narrow bandwidth and is difficult to model since a hole has to be drilled in the substrate and the connector protrudes outside the ground plane, thus not making it completely planar for thick substrates ($h > 0.02\lambda_o$). Also, for thicker substrates, the increased probe length makes the input impedance more inductive, leading to matching problems [9]. It is seen above that for a thick dielectric substrate,which provides broad bandwidth, the microstrip line feed and the coaxial feed suffer from numerous disadvantages. The non-contacting feed techniques which have been discussed below, solve these issues.

Methods of Analysis

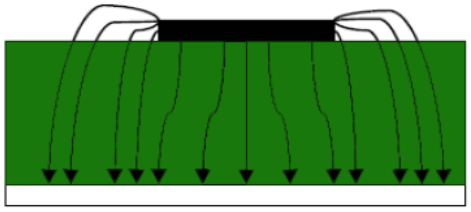
The preferred models for the analysis of Microstrip patch antennas are the transmission line model, cavity model, and full wave model (which include primarily integral equations/Moment Method). The transmission line model is the simplest of all and it gives good physical insight but it is less accurate. The cavity model is more accurate and gives good physical insight but is complex in nature. The full wave models are extremely accurate, versatile and can treat single elements, finite and infinite arrays, stacked elements, arbitrary shaped elements and coupling.These give less insight as compared to the two models mentioned above and are far more complex in nature.

Transmission Line Model

This model represents the microstrip antenna by two slots of width *W* and height *h*, separated by a transmission line of length L. The microstrip is essentially a non-homogeneous line of two dielectrics, typically the substrate and air.



Microstrip Line Figure



Electric Field Lines

Hence most of the electric field lines reside in the substrate and parts of some lines in air. As a result, this transmission line cannot support pure transverse-electricmagnetic(TEM) mode of transmission, since the phase velocities would be different in the air and the substrate. Instead, the dominant mode of propagation would be the quasi-TEM mode. Hence, an effective dielectric constant (ε_{reff}) must be obtained in order to account for the fringing and the wave propagation in the line. The value of ε_{reff} is slightly less then ε_r because the fringing fields around the periphery of the patch are not confined in the dielectric substrate but are also spread in the air as shown in

The expression for ε_{reff} is given by Balanis as:

$$\varepsilon_{reff} = \frac{\varepsilon_r + 1}{2} + \frac{\varepsilon_r - 1}{2}\left[1 + 12\frac{h}{W}\right]^{-\frac{1}{2}}$$

Where

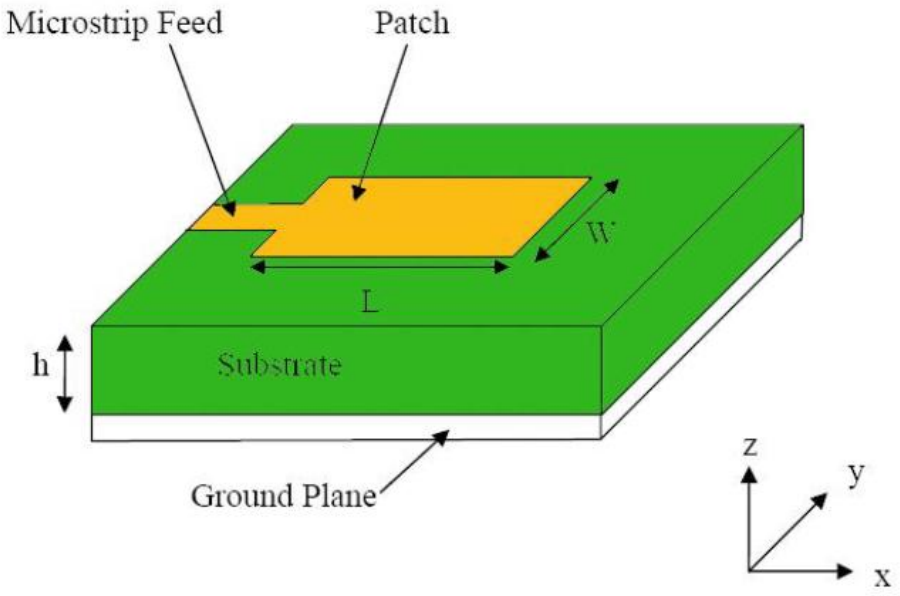
ε_{reff} = Effective dielectric constant

ε_r = Dielectric constant of substrate

h = Height of dielectric substrate

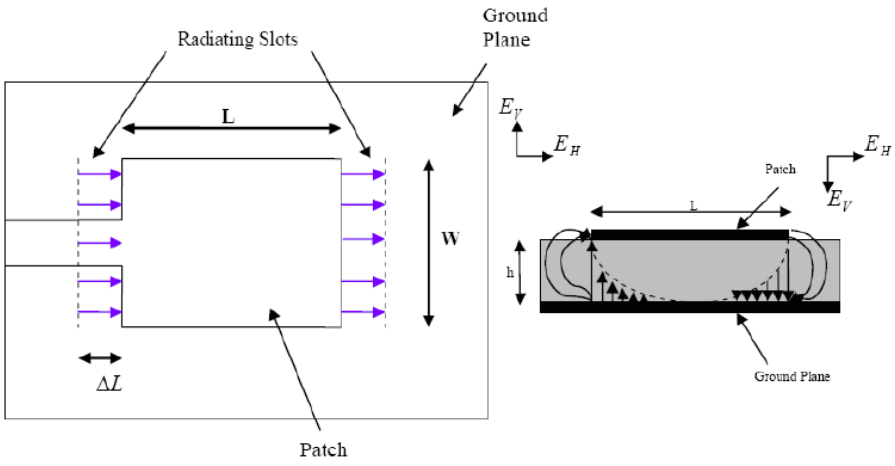
W = Width of the patch

Consider the figure below, which shows a rectangular micro strip patch antenna of length L, width W resting on a substrate of height h. The co-ordinate axis is selected such that the length is along the x direction, width is along the y direction and the height is along the z direction.



Microstrip Patch Antennas

In order to operate in the fundamental TM_{10} mode, the length of the patch must be slightly less than λ/2 where λ is the wavelength in the dielectric medium and is equal to $\lambda_o/\sqrt{\varepsilon reff}$ where λ_o is the free space wavelength. The TM_{10} mode implies that the field varies one λ/2 cycle along the length, and there is no variation along the width of the patch. In the Figure below, the microstrip patch antenna is represented by two slots, separated by a transmission line of length L and open circuited at both the ends. Along the width of the patch, the voltage is maximum and current is minimum due to the open ends. The fields at the edges can be resolved into normal and tangential components with respect to the ground plane.



Top View of Antenna Side View of Antenna

It is seen from Figures that the normal components of the electric field at the two edges along the width are in opposite directions and thus out of phase since the patch is λ/2 long and hence they cancel each other in the broadside direction. The tangential components ,which are in phase, means that the resulting fields combine to give maximum radiated field normal to the surface of the structure. Hence the edges along the width can be represented as two radiating slots, which are λ/2 apart and excited in phase and radiating in the half space above the ground plane.The fringing fields along the width can be modeled as radiating slots and electrically the patch of the microstrip antenna looks greater than its physical dimensions.

The dimensions of the patch along its length have now been extended on each end by a distance ΔL, which is given empirically by Hammerstad as:

$$\Delta L = 0.412h\frac{\left(\varepsilon_{reff}+0.3\right)\left(\frac{W}{h}+0.264\right)}{\left(\varepsilon_{reff}-0.258\right)\left(\frac{W}{h}+0.8\right)}$$

The Effective length of the patch becomes

$$L_{eff} = L + 2\Delta L$$

For a given resonance frequency the effective length is given by

$$L_{eff} = \frac{c}{2f_o\sqrt{\varepsilon_{reff}}}$$

The resonance frequency of the rectangular patch antenna for TM_{mn} is

$$f_o = \frac{c}{2\sqrt{\varepsilon_{reff}}}\left[\left(\frac{m}{L}\right)^2+\left(\frac{n}{W}\right)^2\right]^{\frac{1}{2}}$$

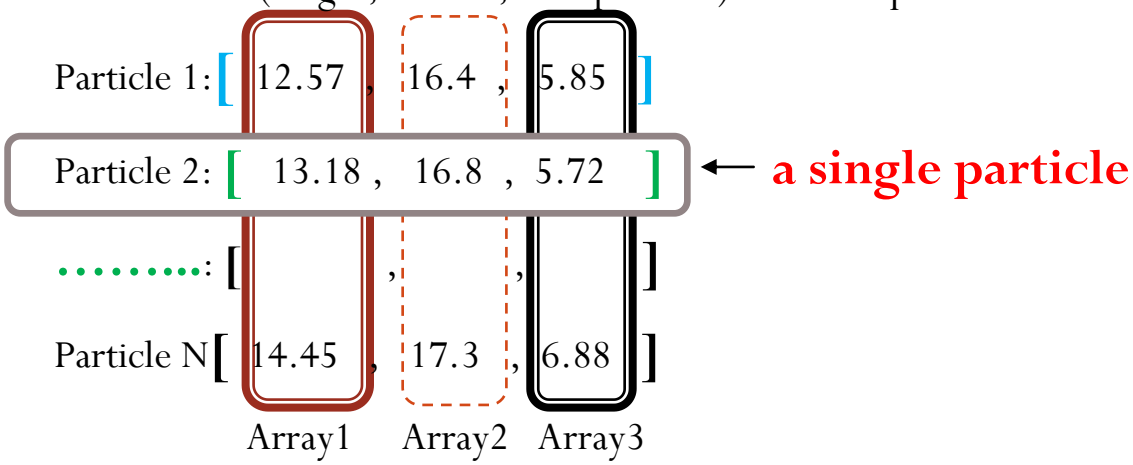
Section-C

- **Implementation Algorithm**
- **Matlab Code and GUI**

- **Implementation Algorithm**

****

Step 1: Each particle in the swarm is identified as a 3 dimension point or a set of 3 numbers(length, width , feed position).For example



Length(l) Width(w) Feed position(fp)

N=number of particles in the swarm

Hence, to create such a swarm an array of numbers is initially chosen each for length, width and feed position.All one array is taken initially.

l =[1 1 1 1 1 1.....1]$_{1Xn}$

w =[1 1 1 1 1 1.....1]$_{1xN}$

fp =[1 1 1 1 1 1.....1]$_{1xN}$

****

Step 2: Upper limit and lower limit of the parameters (l,w,fp) is determined.

Say: wmax ,wmin

lmax , lmin

fpmax , fpmin

****

Step 3:

Initial random distribution of particles in the solution area

for i=1 to N

l(i)= lmin + (lmax-lmin)*random_number;

w(i)= wmin + (wmax-wmin)*random_number;

fp(i)=fmin + (fmax-fmin)*random_number;

END

Particles in the swarm are now randomly distributed in the solution space.

****

Step 4:

With each of the particles at certain 3D co-ordinates, having specific random values for length, width and feed position within their limiting max and min values, the corresponding fitness function value can be calculated. With given height of dielectric, its permittivity and frequency of operation the input impedance of the rectangular patch antenna can be calculated out.Hence for each (l,w,h) ,we can have a Z_0.

Particle 1 : Z_{01}

Particle 2 :Z_{02}

Particle N:Z_{0N}

If 50 ohm is the characteristics impedance of feed ,the absolute difference is the impedance match can be obtained for each of the particle.

Particle 1: $|Z_{01} - 50|$, Particle 2: $|Z_{02} - 50|$, and so on……

****

Step 5:

Now after step 4 we have an array of positive numbers indicating the amount of impedance mismatch .We choose that value of (l,w, fp) as the gbest position for which the positive mismatch value is minimum. Rest other points are pbest position.

$(l_{gbest}, w_{gbest}, f_{gbest})$ = co-ordinate_of (min($|Z_{0k} - 50|$)); for k= 1 to N.

Now we have the gbest position for our 1st iteration.

****

Step 6:

Position Update for each of the particle depending on their own pbest position(current position) and the overall gbest position.

For i=1 to N

$w(i) = w(i) + 2*random\_number*(w_{gbest} - w(i))$;

$fp(i) = fp(i) + 2*random\_number*(fp_{gbest} - fp(i))$; velocity function

$l(i) = l(i) + 2*random\_number*(l_{gbest} - l(i))$;

END

Now we have new position for all the particles .

****

Step 7:

Checking whether the new positions have gone beyond the max min limit for each particle. If so then they are rebounded back within the solution field.

If w(i)>wmax

w(i)=wmax - random_number;

end

So is checked with other particles for wmin ,lmax, lmin,fpmax and fpmin .

****

Step 8:

Plotting of each of the particles in 3 dimensions.

****

Step 9:

Checking the iteration number against the maximum iteration defined.Repeating all the steps from step 4 until the iteration number exceeds the maximum value.

****

Step 10:

The final gbest position co-ordinate (l_{gbest} ,w_{gbest} ,fp_{gbest}) is the optimized anteena design parameters that provides the best impedance match between the characteristics impedance of feed and input impedance of the antenna.

****

• Matlab Code

****

List of MATLAB files:

MAIN.m
MAIN.fig
diffcalc.m

MAIN.m

```
***Default Donotedit-Code for GUI.Detail code not included for this part***

• function varargout = MAIN(varargin)
• function MAIN_OpeningFcn(hObject, eventdata, handles, varargin)
• function varargout = MAIN_OutputFcn(hObject, eventdata, handles)

  function edit1_Callback(hObject, eventdata, handles)
• function edit1_CreateFcn(hObject, eventdata, handles)

  function edit2_Callback(hObject, eventdata, handles)
• function edit2_CreateFcn(hObject, eventdata, handles)

  function edit6_Callback(hObject, eventdata, handles)
• function edit6_CreateFcn(hObject, eventdata, handles)

  function edit9_Callback(hObject, eventdata, handles)
• function edit9_CreateFcn(hObject, eventdata, handles)

  function popupmenu1_Callback(hObject, eventdata, handles)
• function popupmenu1_CreateFcn(hObject, eventdata, handles)
```

```
********************Code to get input from GUI**************************

   #Height of the dielectric
```

- `function HEIGHT_Callback(hObject, eventdata, handles)`
  `handles.metricdata.ht=str2double(get(hObject,'String')) ;`
  `guidata(hObject,handles)`
- `function HEIGHT_CreateFcn(hObject, eventdata, handles)`
  `if ispc && isequal(get(hObject,'BackgroundColor'),`
  `get(0,'defaultUicontrolBackgroundColor'))`
  `set(hObject,'BackgroundColor','white');`
  `end`

```
   #Frequency of Operation
```

- `function Freq_Callback(hObject, eventdata, handles)`
  `handles.metricdata.Freq=str2double(get(hObject,'String')) ;`
  `guidata(hObject,handles)`
- `function Freq_CreateFcn(hObject, eventdata, handles)`
  `if ispc && isequal(get(hObject,'BackgroundColor'),`
  `get(0,'defaultUicontrolBackgroundColor'))`
  `set(hObject,'BackgroundColor','white');`
  `end`

```
#Permitivity of the Dielectric
```

- `function DIELECTRIC_Callback(hObject, eventdata, handles)`
- `function DIELECTRIC_CreateFcn(hObject, eventdata, handles)`

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

#Number of iteration
```

- `function ITRTN_Callback(hObject, eventdata, handles)`

```
handles.metricdata.ITERATION=str2double(get(hObject,'String')) ;
guidata(hObject,handles)
```

- `function ITRTN_CreateFcn(hObject, eventdata, handles)`

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

#Number of swarms
```

- `function SWARM_Callback(hObject, eventdata, handles)`

```
handles.metricdata.SWRM=str2double(get(hObject,'String')) ;
guidata(hObject,handles)
```

- `function SWARM_CreateFcn(hObject, eventdata, handles)`

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
*****************************MAIN Function*****************************
```

- `function START_Callback(hObject, eventdata, handles)`

```
freq=handles.metricdata.Freq;
h=handles.metricdata.ht;
e=get(handles.DIELECTRIC, 'Value');
switch e
    case 1
        er=2.4;
    case 2
        er=2.7;
    case 3
        er=3;
    case 4
       er=3.5;
    case 5
      er=4.2;
    case 6
      er=7;
end


fre=freq;
c=3*10^11;
freq=freq*10^9;
lamda= c/freq;
k=(2*3.14)/lamda;
#defining the upper and lower limits of the design parameters

wmax=lamda/2;
wmin=lamda/3;
#maximum and minimum value of width of the patch

erefflmx=((er+1.0)/2.0) +  (er-1)/(2*sqrt(1.0+12.0*(h/wmax)));
dllmx=(0.412*h*((erefflmx+0.3)*((wmax/h)+0.264)))/((erefflmx-0.258)*((wmax/h)+0.8));
lmin=(lamda/(2*sqrt(erefflmx)))- (2*dllmx);
erefflm=((er+1.0)/2.0) +  (er-1)/(2*sqrt(1.0+12.0*(h/wmin)));
dllm=(0.412*h*((erefflm+0.3)*((wmin/h)+0.264)))/((erefflm-0.258)*((wmin/h)+0.8));
lmax=(lamda/(2*sqrt(erefflm)))- (2*dllm);
#maximum and minimum value of length of the patch

gmx=(wmax*( 1- (((k*h)^2))/24))/(120*lamda);
rinmx=1/(2*gmx);
fmax=(acos(sqrt(50/rinmx))*lmax)/3.14;
gmn=(wmin*( 1- (((k*h)^2))/24))/(120*lamda);
rinmn=1/(2*gmn);
fmin=(acos(sqrt(50/rinmn))*lmin)/3.14;
#maximum and minimum value of the feed position

n=handles.metricdata.ITERATION;
s=handles.metricdata.SWRM;
#n is the number of iterations and s equals number of swarms
```

```
l=ones(1,s);
w=ones(1,s);
fp=ones(1,s);
#Creating Swarms

d=rand(1,s);
u=rand(1,s);
o=rand(1,s);
csvwrite('C:\RAND1.txt',d);
csvwrite('C:\RAND2.txt',u);
csvwrite('C:\RAND3.txt',o);


for i=1:s
    l(i)= lmin + (lmax-lmin)*d(i);
    w(i)= wmin + (wmax-wmin)*u(i);
    fp(i)=fmin + (fmax-fmin)*o(i);
end
#Initial random distribution of swarms


z=ones(1,s);
u=ones(1,n);
q=ones(1,n);
k=1;

for j=1:n
     for i=1:s
     z(i)=diffcalc(l(i),w(i),fp(i),h,fre); #function diffcalc defined
later
     end
  u(j)=min(z);
 [a,b]=min(z);
    q(j)=(sum(z)/s)+50;
  A=w(b);
  B=fp(b);
  C=l(b);

   if j < (n)
     for i=1:s
      w(i) = w(i) + 2*rand(1)*(A-w(i));
      fp(i)= fp(i) + 2*rand(1)*(B-fp(i));
      l(i)=l(i) + 2*rand(1)*(C-l(i));
  end

#velocity function

  for i=1:s

      if w(i)>wmax
          w(i)=wmax-rand(1);
      end
      if w(i)<wmin
          w(i)=wmin+rand(1);
      end
```

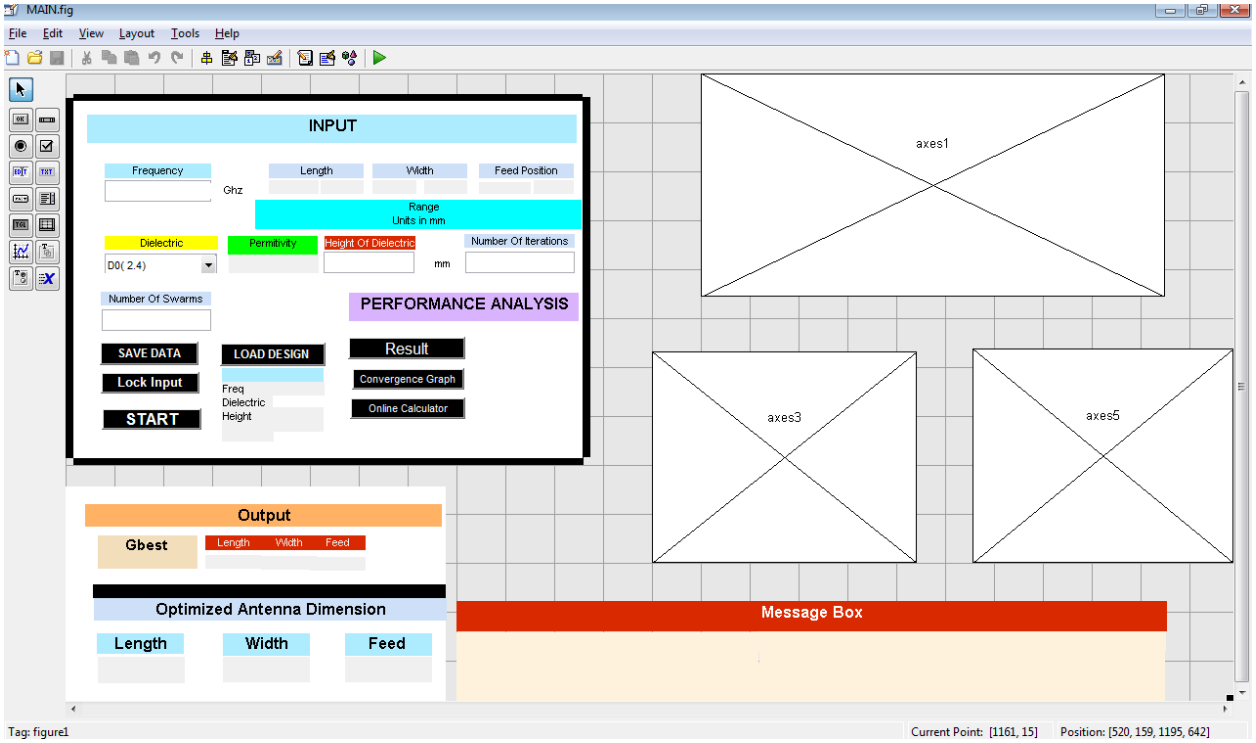
```
        if fp(i)>fmax

            fp(i)=fmax-rand(1);
        end

        if fp(i)<fmin
             fp(i)=fmin+rand(1);
        end

        if l(i)<lmin
             l(i)=lmin+rand(1);
        end

         if l(i)>lmax
             l(i)=lmax-rand(1);
        end


  end
end

#Rebounce Functions

axes(handles.axes1);
cla;
plot3(l,w,fp,'*');
xlim([lmin,lmax]);
ylim([wmin,wmax]);
zlim([fmin,fmax]);
grid on;
pause(1);

#3D plot
end

axes(handles.axes3);
cla;
r=1:1:n;
plot(r,u,'--rs','LineWidth',2,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',3)
xlabel('Iteration Number');
ylabel('Fitness Function ');
grid on;

#2D plot of fitness function vs Iteration
axes(handles.axes5);
cla;
r=1:1:n;
plot(r,q,'--rs','LineWidth',2,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',3)
```

```
   xlabel('Iteration Number');
   ylabel('Average Impedance ');
   grid on;

  #2D plot of average impedance vs Iteration

   set(handles.LR,'String',l(b));
   set(handles.WR,'String',w(b));
   set(handles.FR,'String',fp(b));
 #Display of the result
   zx=[ l(b) w(b) fp(b)];
   csvwrite('C:\z.txt',zx);
 #Storing of the data
```

******Additional function for GUI.Detail program not included.*******

- function LOADDESIGN_Callback(hobject,eventdata,handles)
- function SAVEDATA_Callback(hObject, eventdata, handles)
- function CHECKINPUT_Callback(hObject, eventdata, handles)
- function pushbutton9_Callback(hObject, eventdata, handles)
-

MAIN.fig



Diffcalc.m

```
function [q] = diffcalc(l,w,f,h,freq)
c=3*10^11;
freq=freq*10^9;
lamda=c/freq;
k=(2*3.14)/lamda;
g=(w*( 1- (((k*h)^2))/24))/(120*lamda);
rin=1/(2*g);
r=rin*(cos((3.14*f)/l)^2);
q=abs(r-50); #Fitness Function
end
```

NOTE:

- **Few unnecessary line of codes are omitted to shorten the overall length of the matlab code.**
- **Matlab files are compiled and converted to .exe file and installers are generated.**
- **The software can now run as a standalone application program in any platform.**

Matlab Code on execution : GUI

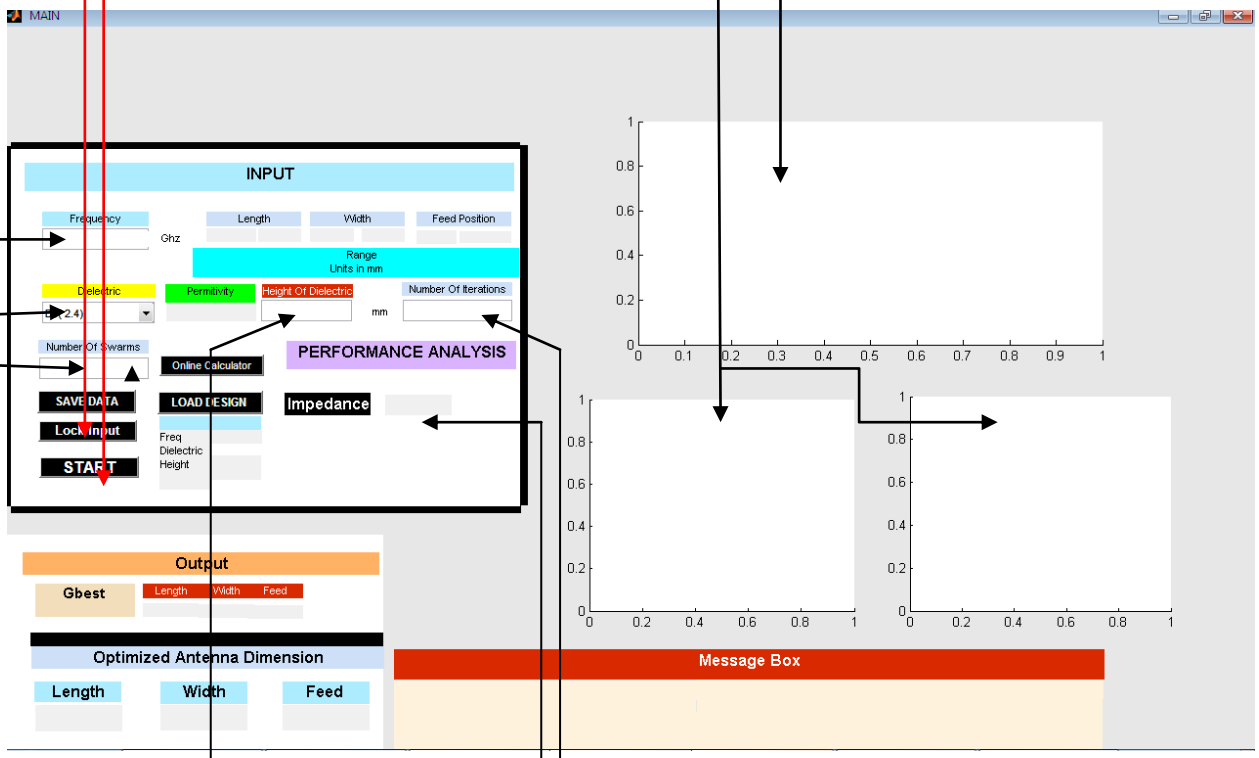
GUI:MAIN

Saving data

Fitness function convergence

Start of the execution

Solution Field.



Input:
swarms

Input:
Height of dielectric

Input:
Number of iterations

Input:
Permittivity Of the dielectric

Final Impedance

Input:
Frequency of operation

Section-D

- ❖ **Results**
- ❖ **Simulation Process in IE3D.**

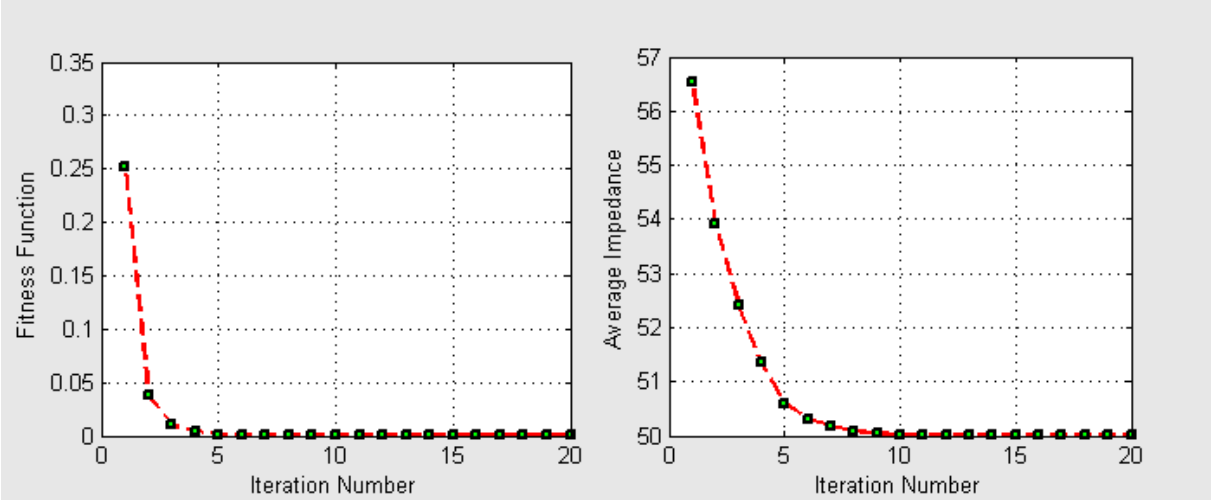
- **Results and IE3D simulation**

1

`

INPUT

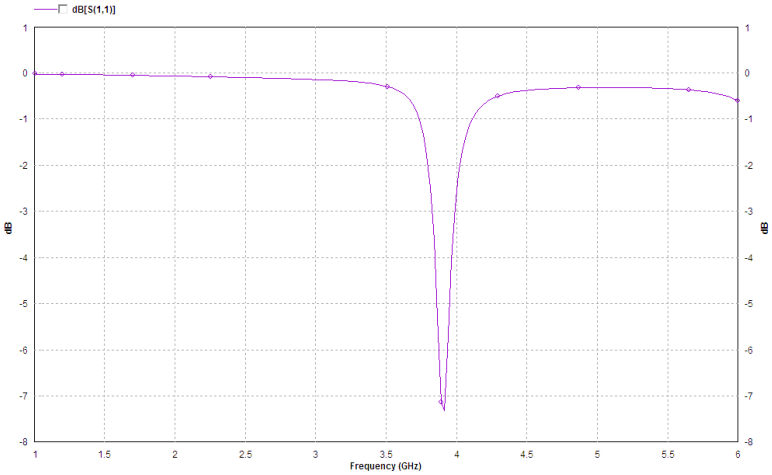
• Frequency	4 GHz
• Height of Dielectric	1.524 mm
• Permittivity	2.4
• Iteration	20
• Swarms	75



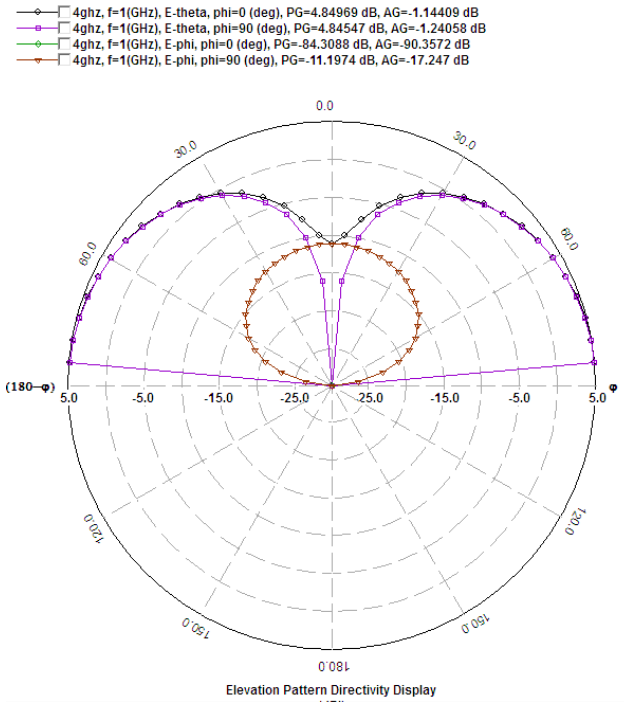
Optimized Result:

Length	=	23.46	mm
Width	=	28.2373	mm
Feed	=	8.811	mm
Input Impedance	=	50	ohm

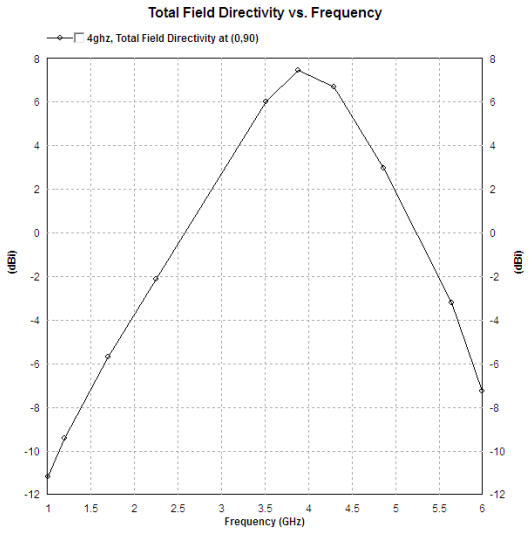
Simulation Result :



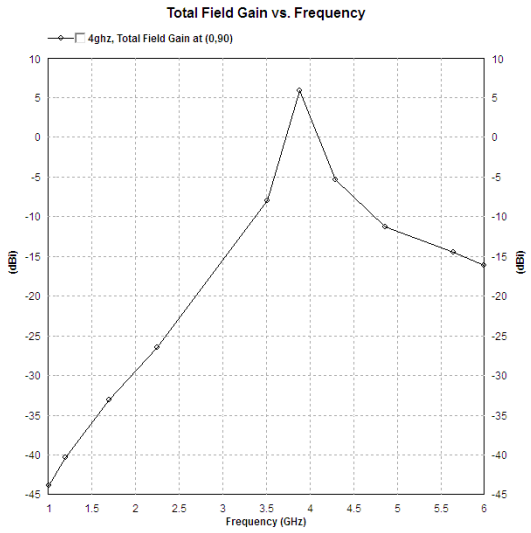
Return loss vs Frequency



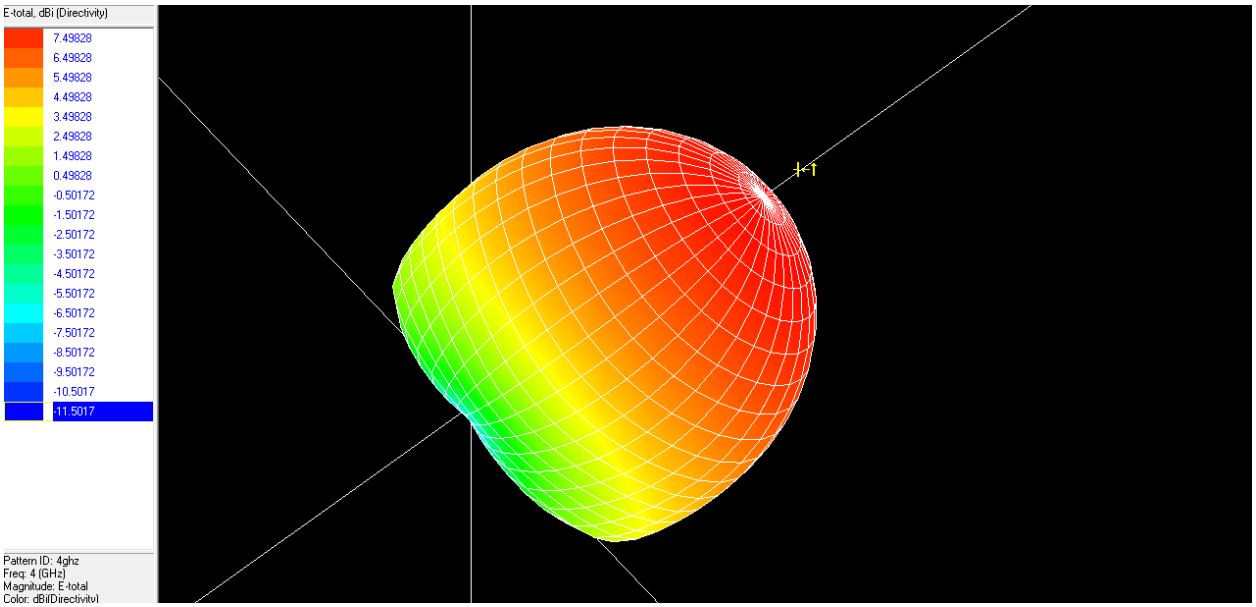
2d pattern plot



Directivity vs. freq



Gain vs. freq

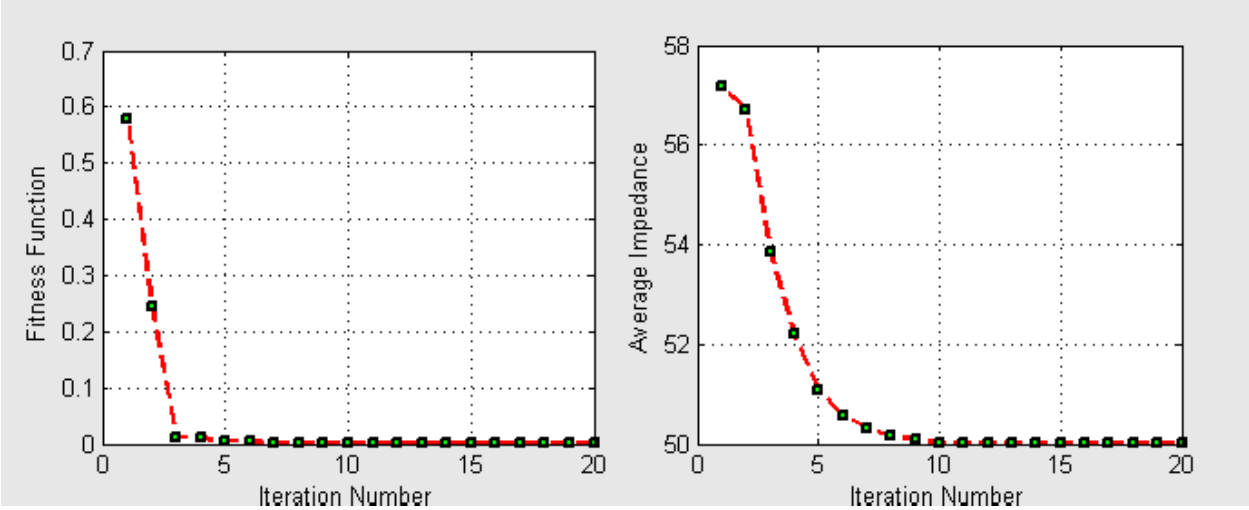


3D radiation plot at 4 Ghz

2

INPUT

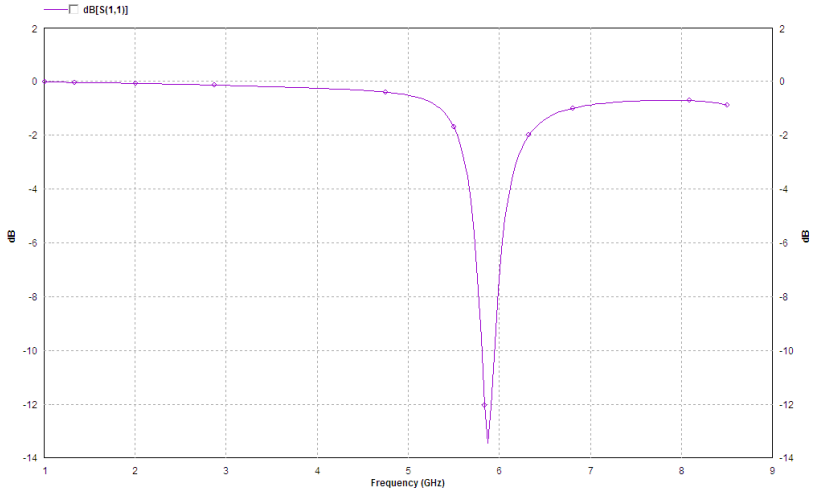
• Frequency	6 GHz
• Height of Dielectric	1.524 mm
• Permittivity	2.4
• Iteration	20
• Swarms	75



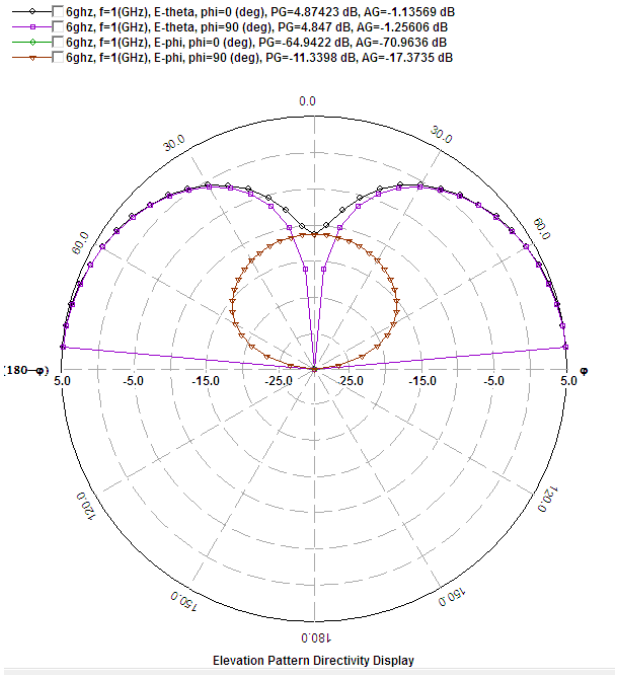
Optimized Result:

Length	=	15.2534 mm
Width	=	19.4515 mm
Feed	=	4.69055 mm
Input Impedance	=	50 ohm

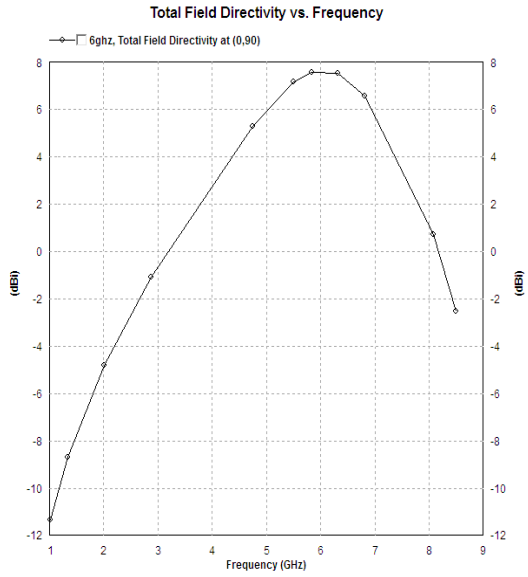
Simulation Result :



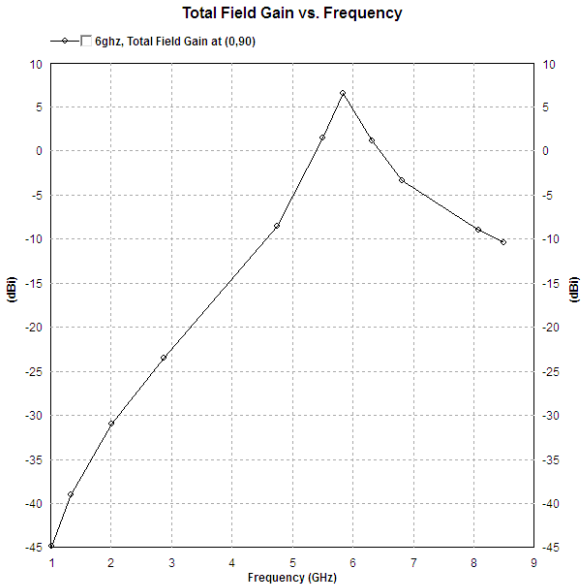
Return loss vs. Frequency



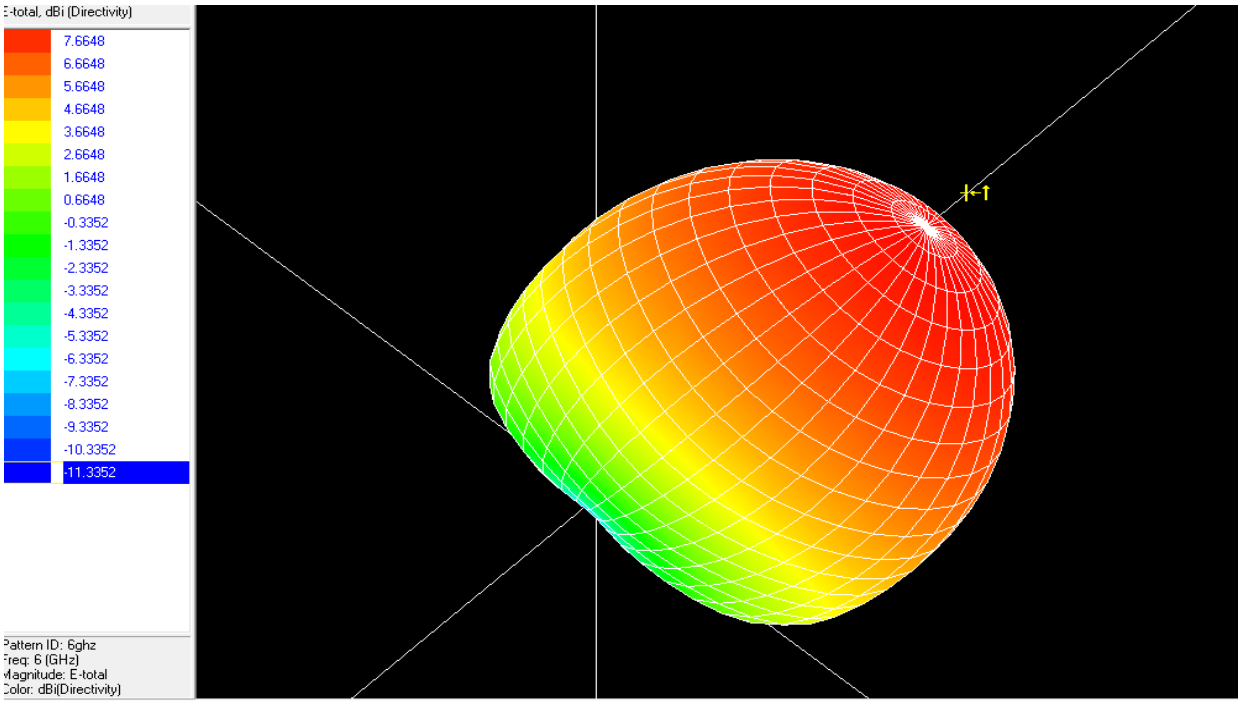
2D pattern plot



Directivity vs. freq



Gain vs. Freq

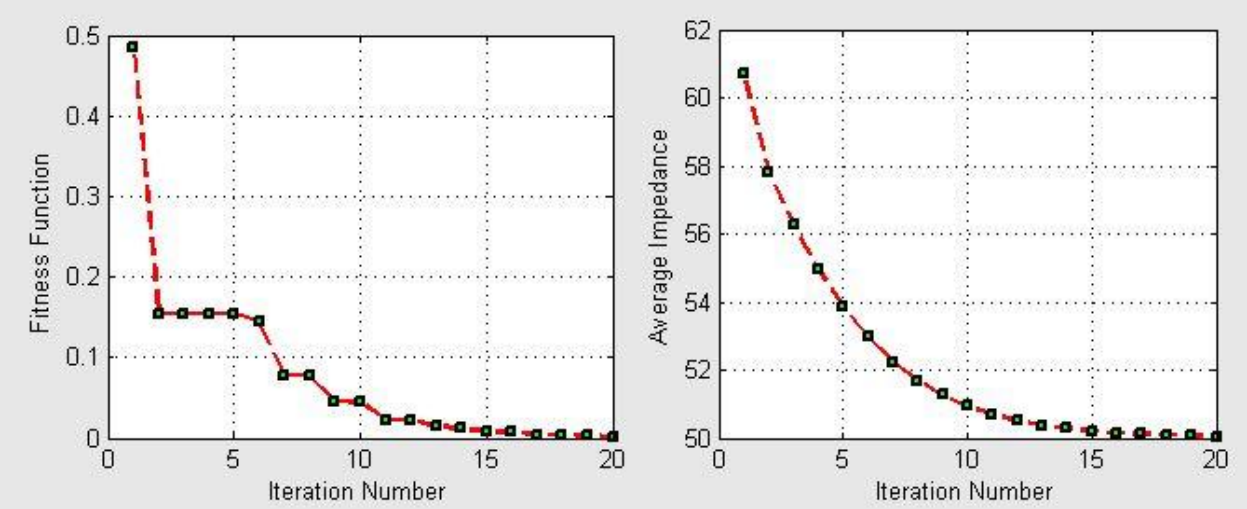


3D Radiation plot at 6 Ghz

3

INPUT

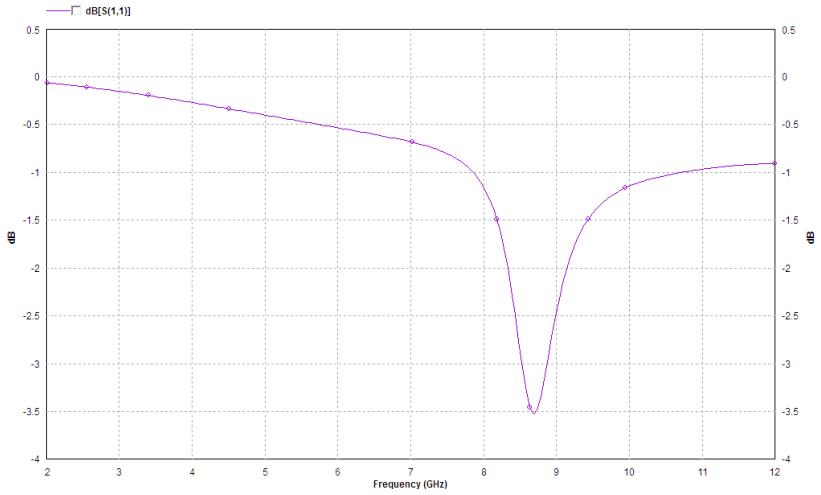
• Frequency	9 GHz
• Height of Dielectric	1.5 mm
• Permittivity	2.4
• Iteration	20
• Swarms	75



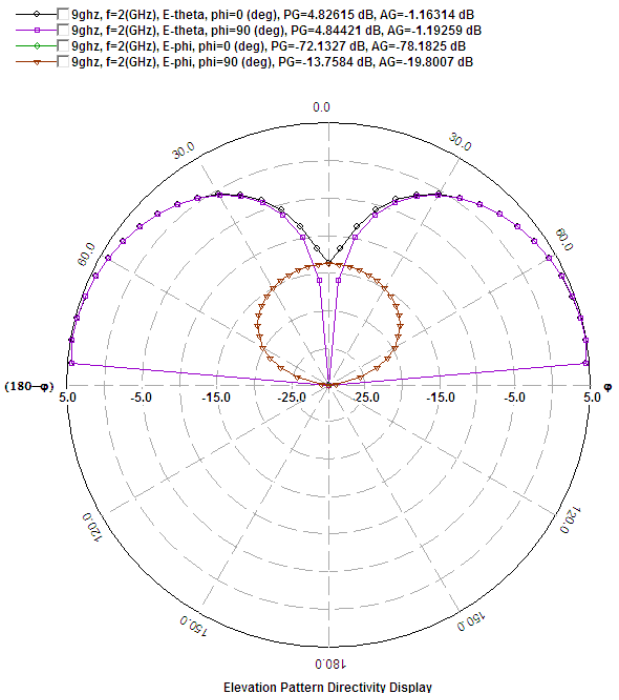
Optimized Result:

Length	=	9.85667	mm
Width	=	12.2143	mm
Feed	=	3.8463	mm
Input Impedance	=	50	ohm

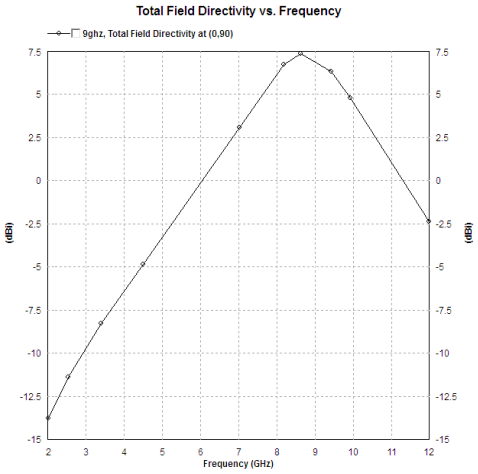
Simulation Result :



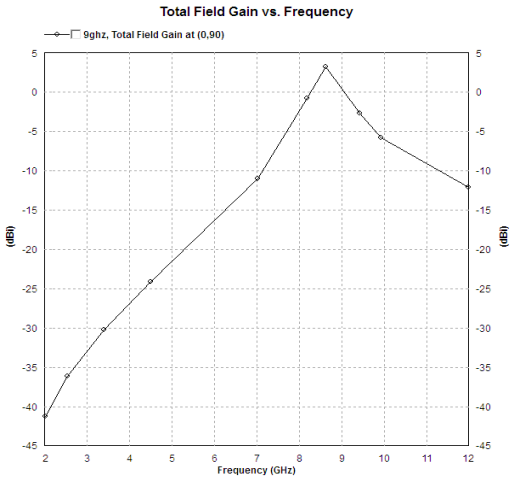
Return loss vs. Frequency



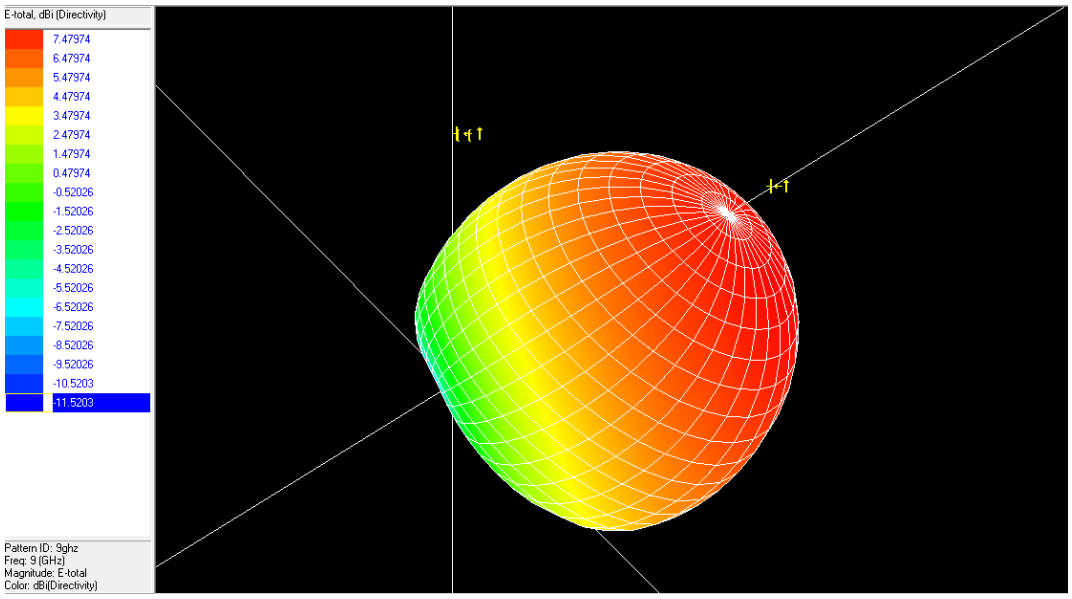
2D pattern display



Directivity vs. Freq



Gain vs Freq

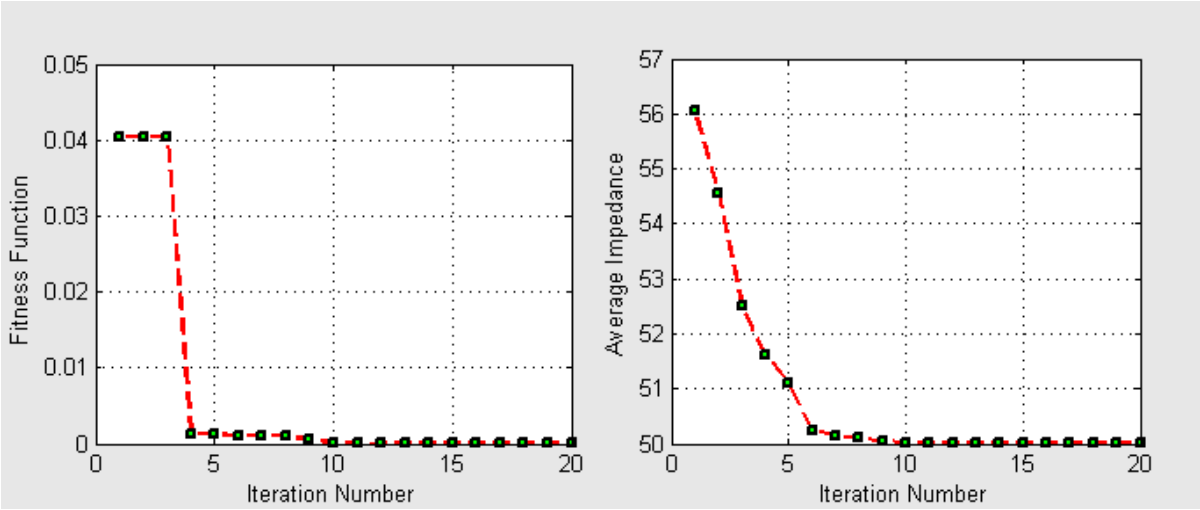


3d Radiation plot for 9 Ghz

4

INPUT

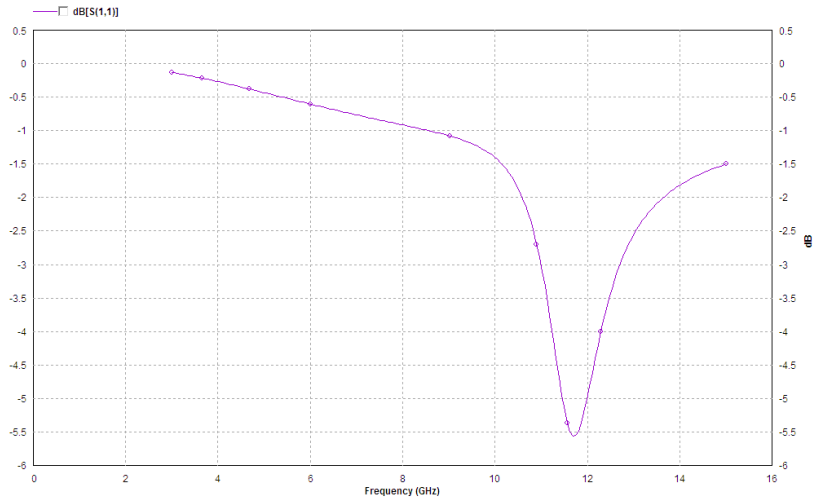
• Frequency	12 GHz
• Height of Dielectric	1.524 mm
• Permittivity	2.4
• Iteration	20
• Swarms	75



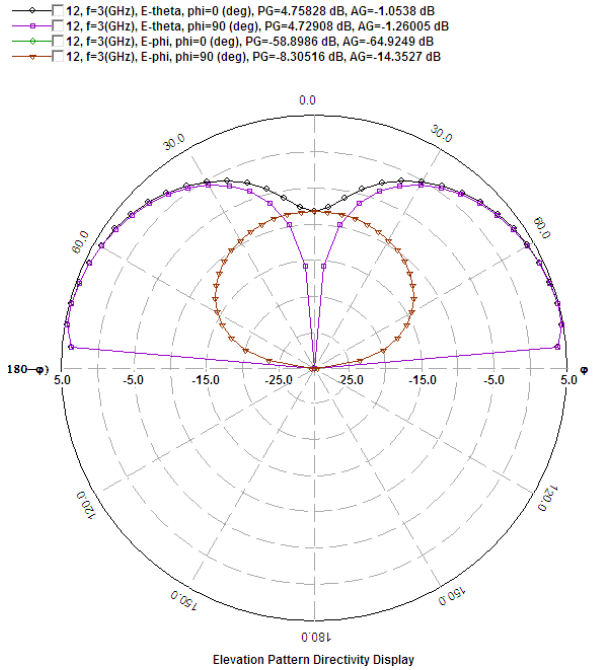
Optimized Result:

Length	=	7.08529 mm
Width	=	9.51941 mm
Feed	=	2.19892 mm
Input Impedance	=	50 ohm

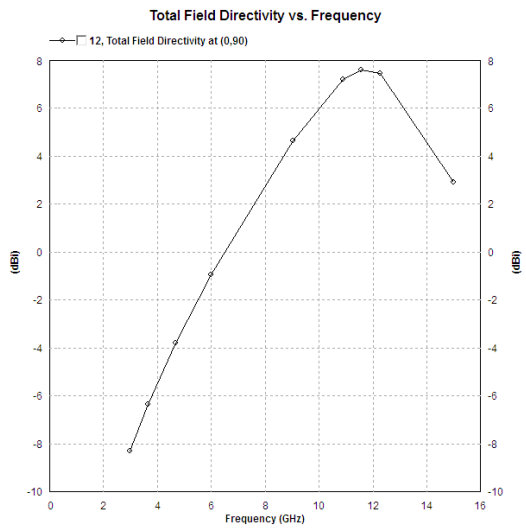
Simulation Result :



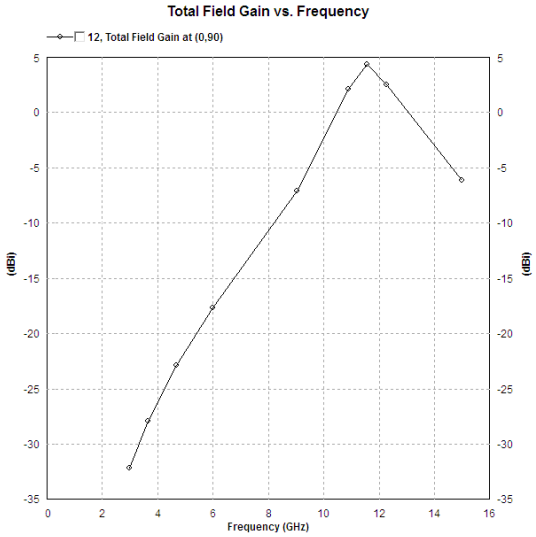
Return loss vs Frequency



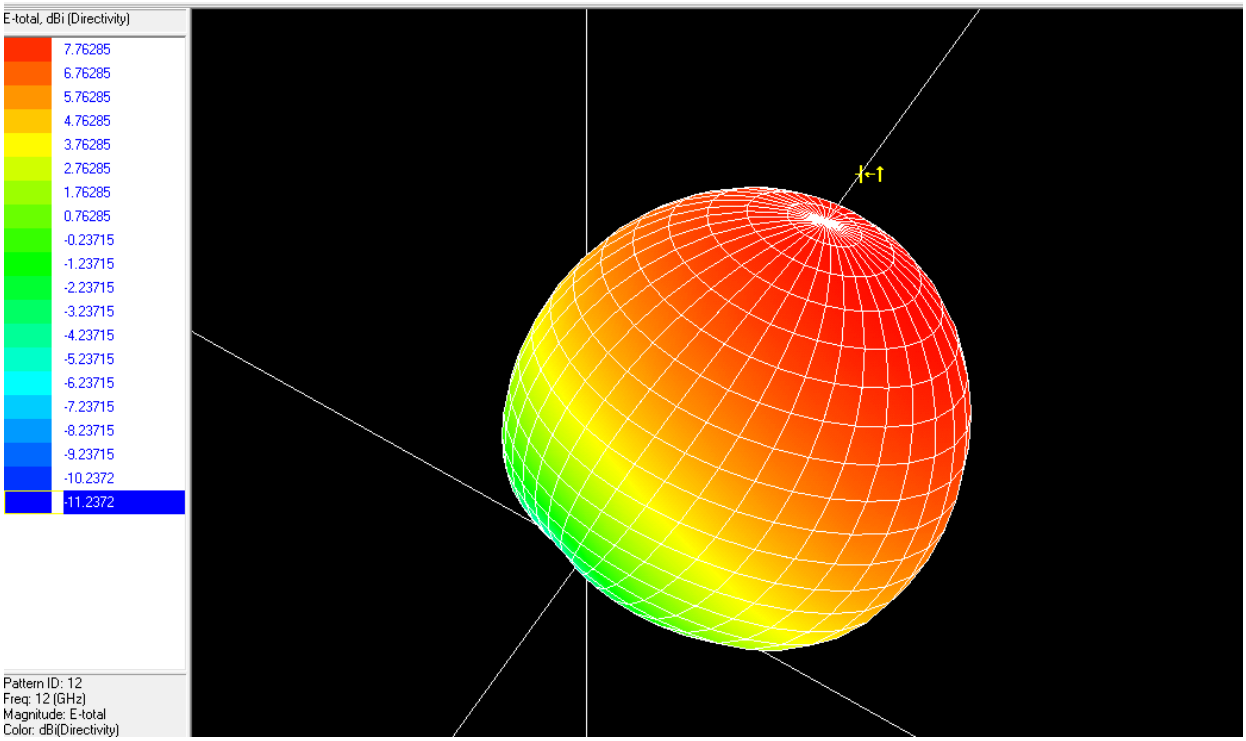
2D pattern Plot



Directivity vs freq



Gain vs Freq

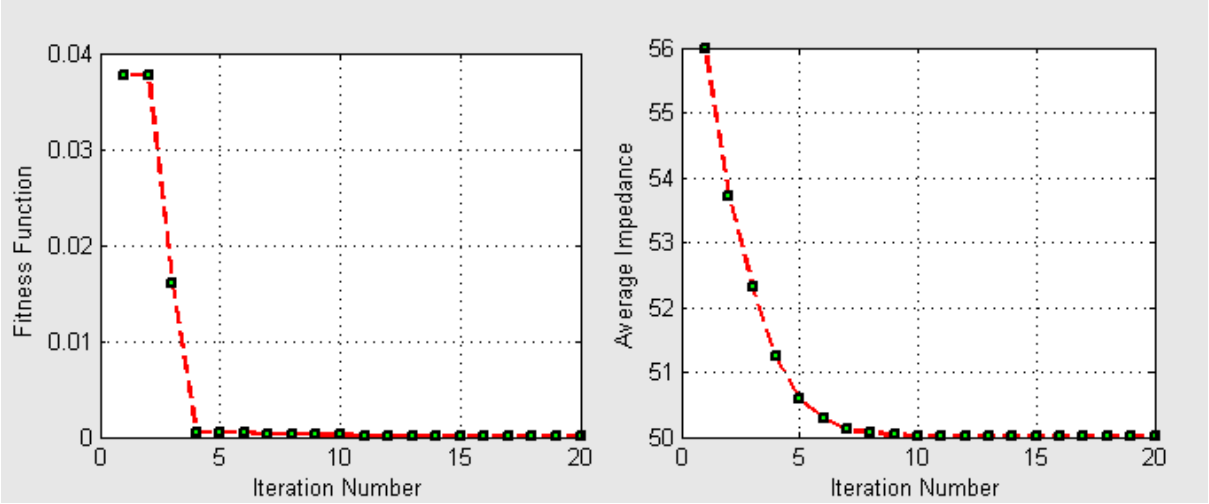


3D Radiation pattern for 12Ghz

6

INPUT

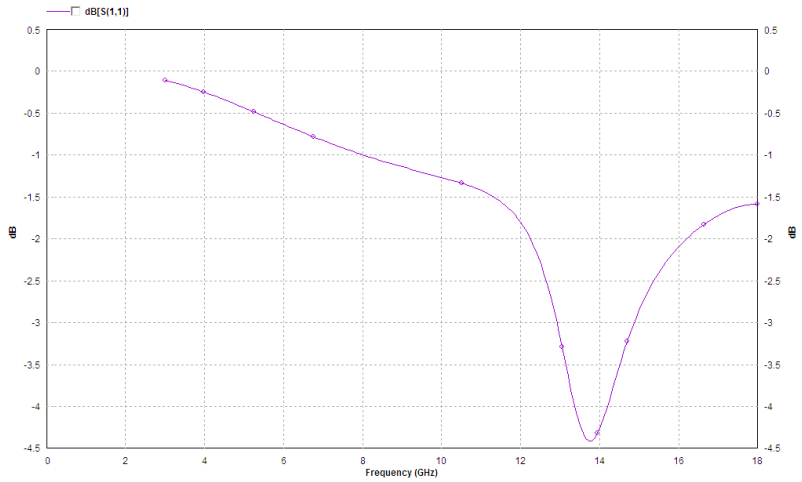
• Frequency	14 GHz
• Height of Dielectric	1.524 mm
• Permittivity	2.4
• Iteration	20
• Swarms	75



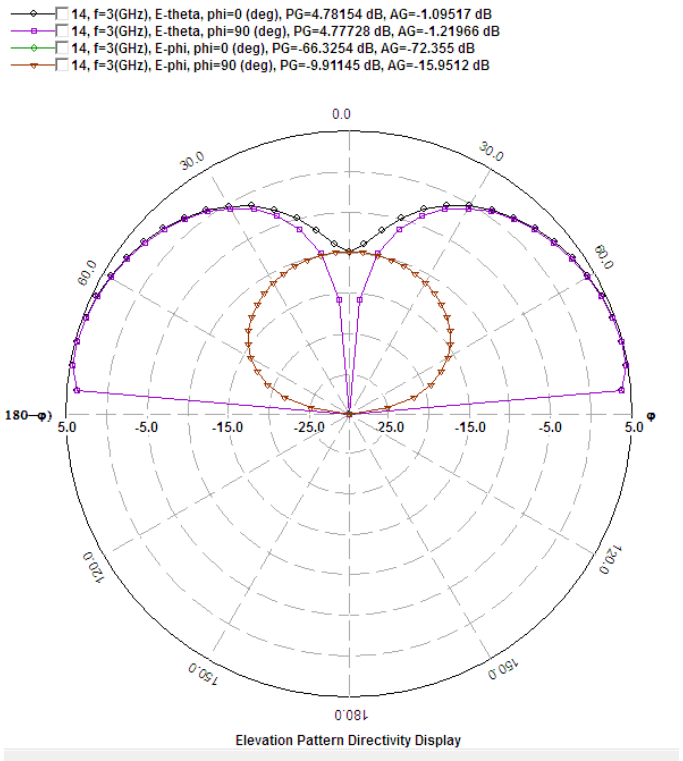
Optimized Result:

Length	=	5.88165 mm
Width	=	8.30229 mm
Feed	=	1.8157 mm
Input Impedance	=	50 ohm

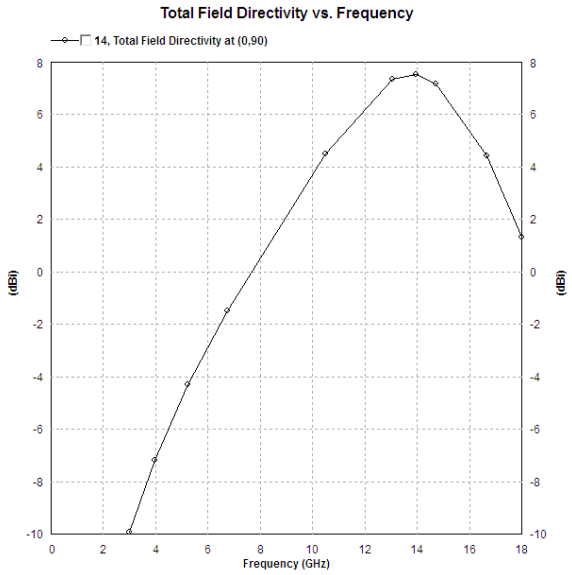
Simulation Result :



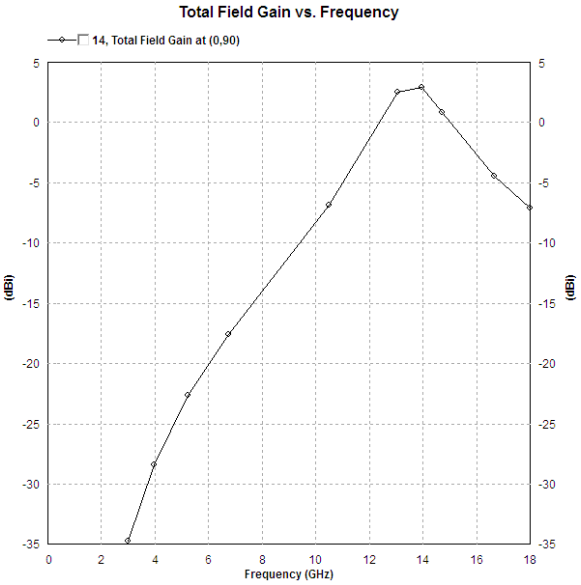
Return loss vs Frequency



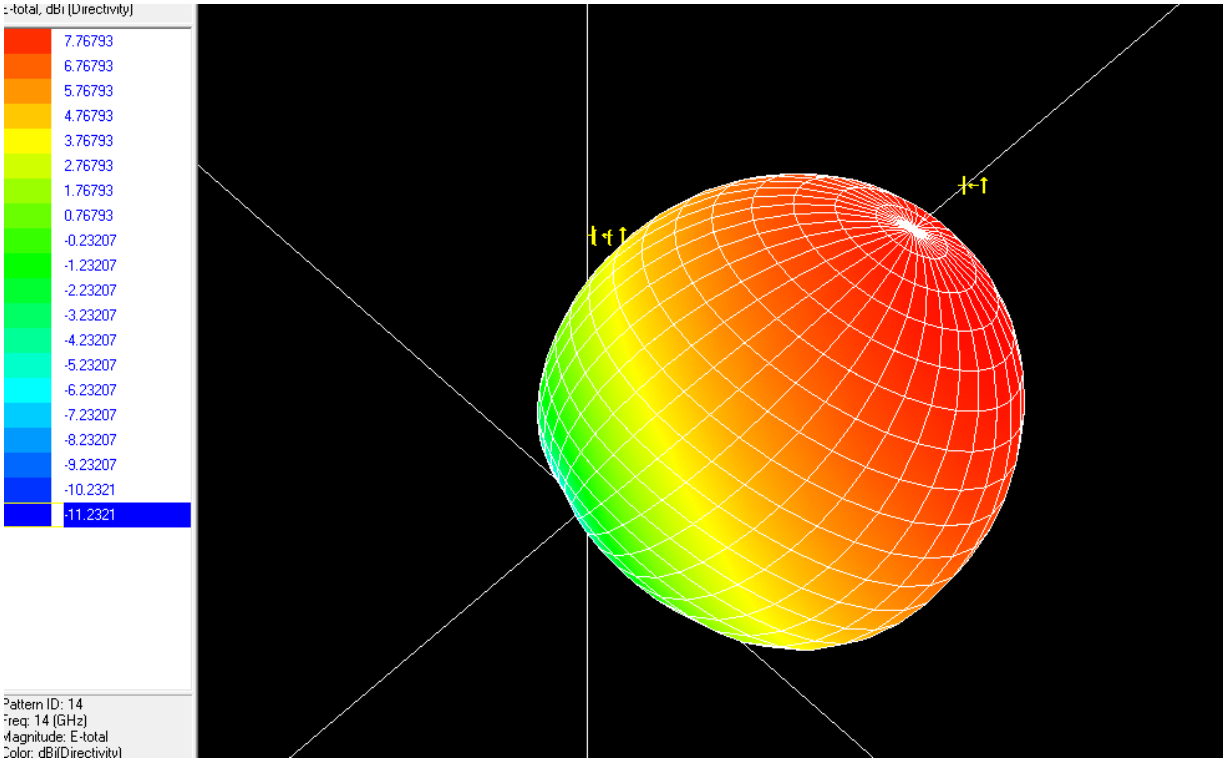
2D pattern Plot



Directivity vs. freq



Gain vs. Freq



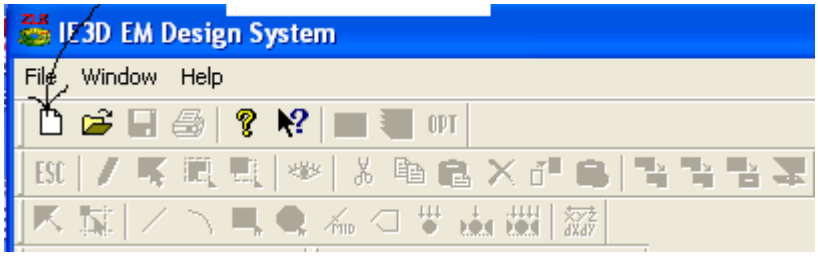
3d Radiation plot for 14 Ghz

Section –E

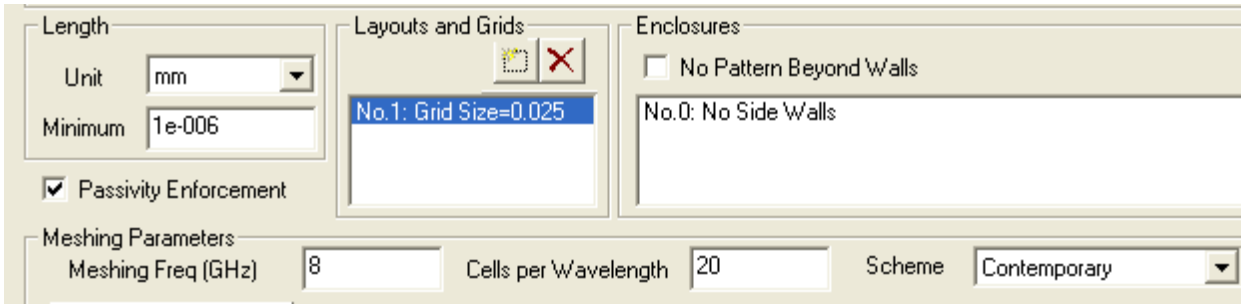
- Simulation Process

Simulation Procedure using IE3D

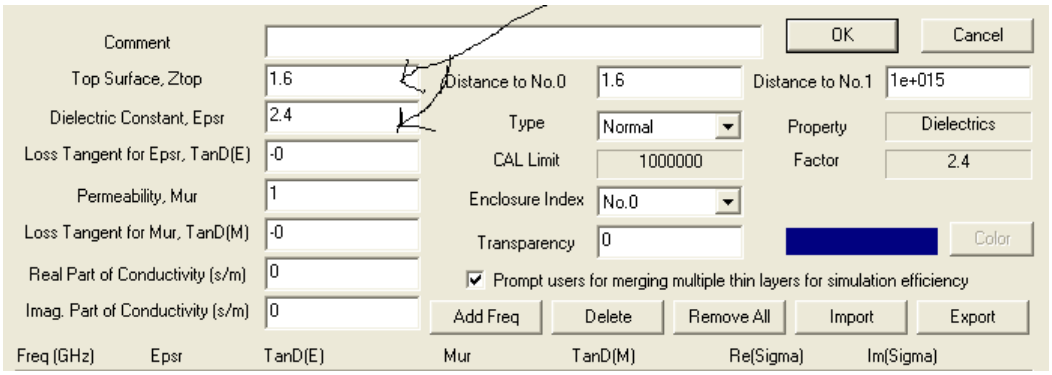
STEP1: Run MGRID. File->New command. MGRID shows the Basic Parameters dialog.



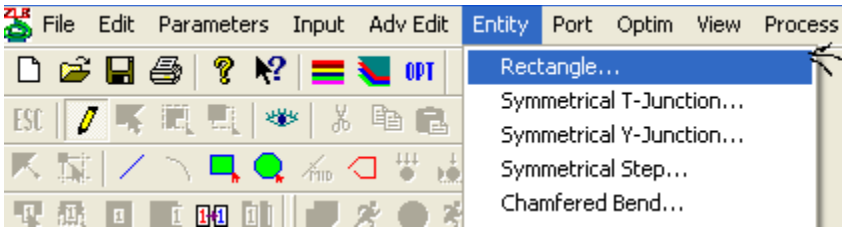
STEP2: Unit is choosen as "mm". In the Meshing Parameters group,



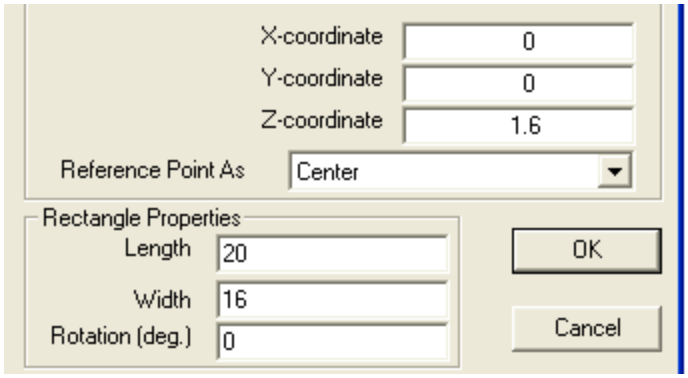
STEP3: New substrate is clicked. Top Surface, Ztop=1.524. Dielectric Constant, Epsr = 2.4 ,loss tan= 0.001



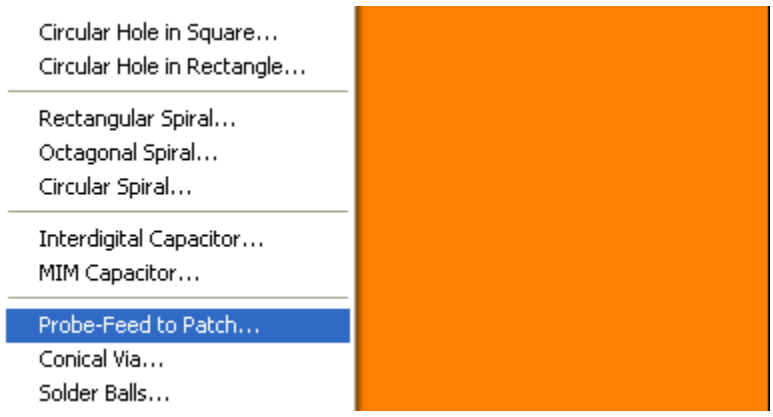
STEP4: We build a rectangle . Entity->Rectangle. MGRID will prompt the parameters.



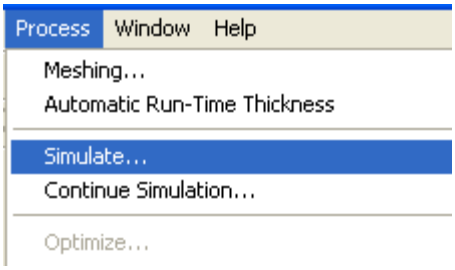
STEP5: Length and Width is taken as input.



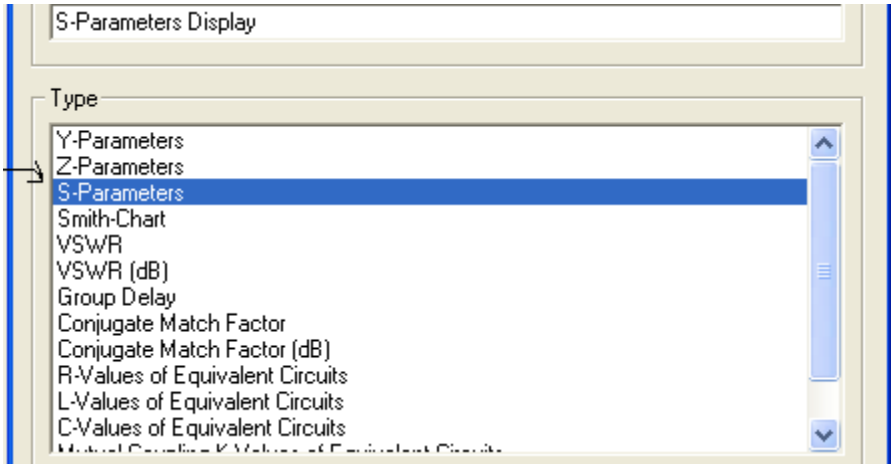
STEP6: Now we have to select probe feed to patch from entity.



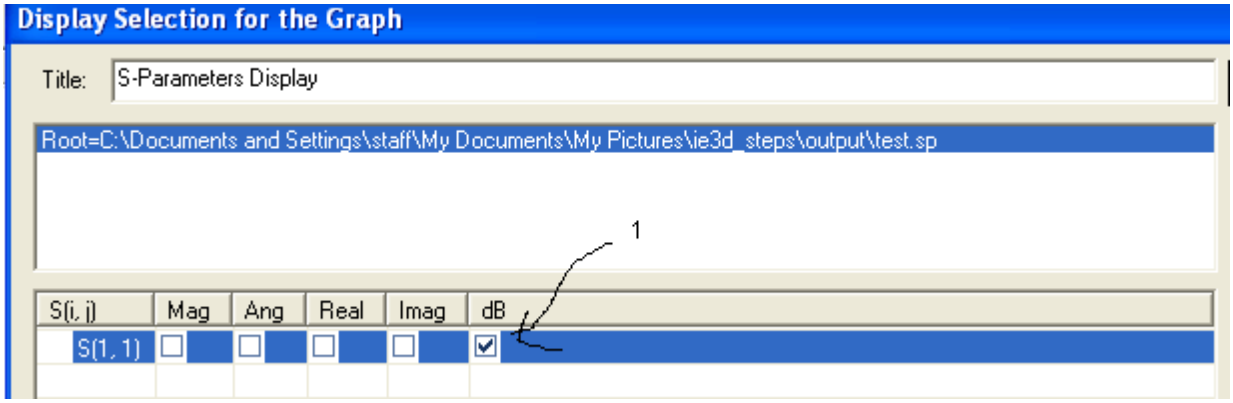
STEP7: Then we have to save the format to do simulation .Then the ultimate step is simulation. Select Process->Simulate command. The Simulation Setup dialog



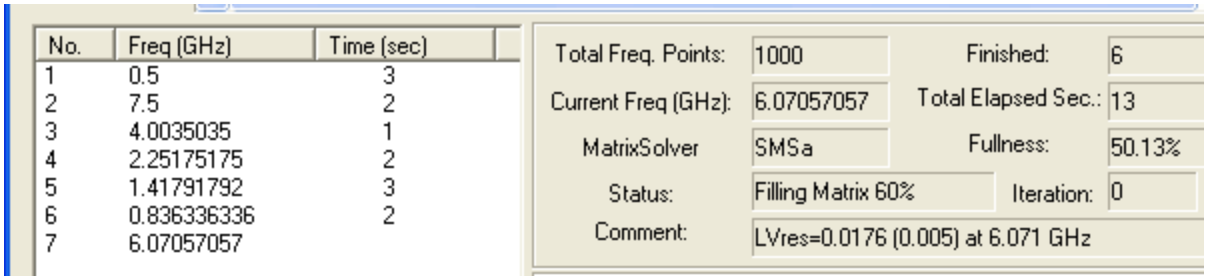
STEP8: We have to define the parameter of the graph.



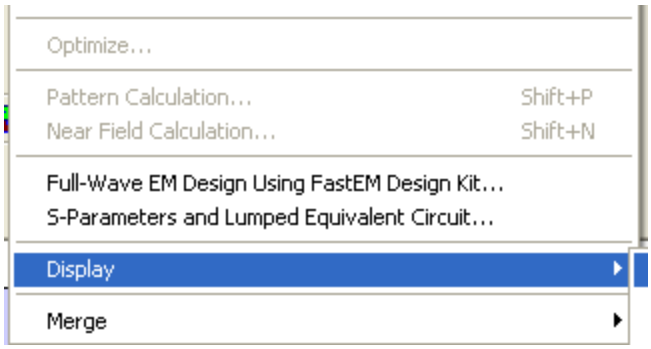
STEP9: We have to define the parameter of display.



Step10: Now the following pic shows . MGRID will invoke IE3D to perform the simulation in the background. It takes seconds to finish.



Step11: To see s-parameter graph we have to select display >> s-parameter.



Step12: To get the 2d and 3d radiation pattern ,.pat file has to be checked and created when simulation box appears.

Section-F

❖ **Conclusion**

Conclusion:

Microstrip patch antenna radiate primarily because of the fringing fields between the patch edge and the ground plate.for a good performance of the antenna a thick dielectric substrate having a low dielectric constantis necessary since it provides larger bandwidth,better radiation and better efficiency.However it leads to a larger antenna size.In order to reduce the size of the micro strip antenna substrates with higher dielectric constant must be used which are less efficient and result in narrow bandwidth.hence a trade-off must be realized between antenna performance and antenna dimensions.