Using Particle-Swarm Optimization for Antenna Design

Magnus Olofsson

Luleå University of Technology
MSc Programmes in Engineering
Department of Computer Science and Electrical Engineering
EISLAB
Using Particle-Swarm Optimization for Antenna Design

Magnus Olofsson

Luleå University of Technology
Dept. of Computer Science and Electrical Engineering
EISLAB

17th February 2006
From a historical perspective, electromagnetic modelling and its techniques of optimization are relatively new to the academic community. Their existence has facilitated the development of complex electromagnetic structures, and provides invaluable aid when designing electronic products that face strict radiation legislation. This thesis provides an introduction to electromagnetic modelling using the Partial element equivalent circuit (PEEC) method, and an in-depth description and evaluation of the technique of particle-swarm optimization.

PEEC is a current research topic at the Embedded Internet Systems Laboratory (EISLAB). This approach describes electromagnetic models and couplings by equivalent circuits. It arises from inductance calculations and allows for inclusion of lumped elements describing voltage sources, resistances, inductances, and capacitors. As a direct result of the research, a local electromagnetic solver is available.

The main objective is to merge the existing EISLAB solver with the particle-swarm algorithm, for optimizing various electromagnetic structures. A technique for calculating the radiated field from the models is also discussed and used for the purpose of optimizing a dipole array.

Particle-swarm optimization was developed in 1995 and models the movement and intelligence of swarms. Behind the algorithm are a social psychologist and an electrical engineer, who developed the optimizer inspired by nature. The technique has proven successful for many electromagnetic problems and is a robust and stochastic search method. The optimization algorithm alters the input file to the PEEC solver, thus affecting the physical description of the electromagnetic structures, and evaluates the result that is returned from the solver.

The particle-swarm algorithm worked well on several problems. It was used for optimizing mathematical functions and electromagnetic problems. The optimized antennas were determined to have desired resonant frequencies, high gain, and low weight and return losses. The patch antennas turned out to be troublesome to handle, thus some improvements such as inclusion of ground planes, are discussed.
Finding a thesis proposal that suggested high academic level, involving complex optimization of electromagnetic problems, triggered a desire to carry out my Master’s thesis at the Embedded Internet Systems Laboratory, Luleå University of Technology.

The work surrounding the thesis puts optimization of electromagnetic problems in a new light for me. I expect that some time still remains before a generic easy-to-use electromagnetic solver exists on the market, especially with a built-in optimizer. Despite the debate whether there really exists a need for such a solver, I hope that my work will be a contribution in taking some local research projects toward optimization.

I take pleasure here in thanking Dr. Jonas Ekman, who has been very helpful and supportive. Additional gratitude goes to David and Krister who contributed to the improvement of the binary implementation of the optimization algorithm.

Magnus Olofsson
# CONTENTS

**CHAPTER 1: INTRODUCTION**  
1

**CHAPTER 2: INTRODUCTION TO ELECTROMAGNETIC MODELLING USING PEEC**  
2.1 Background .................................................. 3  
2.2 Introduction to PEEC .......................................... 5  
2.3 Derivation of basic PEEC theory ............................. 5  
2.4 EISLAB’s PEEC solvers ...................................... 9

**CHAPTER 3: PARTICLE-SWARM OPTIMIZATION**  
3.1 Background .................................................. 11  
3.2 Genetic algorithms .......................................... 11  
3.3 Theory ....................................................... 12  
3.4 Discrete binary representation .............................. 16

**CHAPTER 4: ELECTROMAGNETIC FIELD AND ANTENNA PARAMETERS**  
4.1 The spherical coordinate system ............................ 19  
4.2 Electromagnetic field from a Hertzian dipole .............. 20  
4.3 Antenna parameters ........................................ 21  
4.4 Verification of method ..................................... 23

**CHAPTER 5: IMPLEMENTATION**  
5.1 Building blocks ............................................. 25  
5.2 Optimization of a known function .......................... 27

**CHAPTER 6: RESULTS**  
6.1 Antenna optimization ...................................... 31

**CHAPTER 7: CONCLUSIONS AND FURTHER WORK**  
7.1 Conclusions ............................................... 37  
7.2 Further work .............................................. 37

**APPENDIX A: MATLAB CODE**  
39
Electromagnetic interference (EMI) and electromagnetic compatibility (EMC) play a very important role in the society of today. While hardware overall decreases in size as frequencies increase, a major part of the electrical systems starts to act as radiating and receiving antennas.

Traditionally, SPICE and its counterparts have been used to model the electrical behavior of the systems, without the consideration of electromagnetic couplings. To estimate the latter, cumbersome manufacturing of experimental models based on experience and rules of thumb preceded measurements in shielded chambers. Since the manufacturer of incompatible devices may suffer from large fines due to the strict EMC regulations [1], the use of electromagnetic modelling and its techniques of optimization are important tools for avoiding post-production measures. The need for experimental models, is likely to decrease or be eliminated, thus saving significant amounts of time and money.

At EISLAB, the motivation for using PEEC is mainly its use of equivalent circuits. This technique is used for modelling of various physical problems, such as mechanical, thermodynamic, and ultrasonic models. The desire is to handle all types of problems as one model in one solver.

This thesis is the first step of the far-reaching objective to incorporate an optimization algorithm in EISLAB’s PEEC solver.
CHAPTER 2

Introduction to electromagnetic modelling using PEEC

2.1 Background

Maxwell’s equations form the backbone of electromagnetic theory. James Clerk Maxwell (1831 - 1879) was a Scottish mathematical physicist who reduced the empirical and theoretical knowledge of electricity and magnetism into a set of equations. Maxwell realized that only four\(^1\), at that time existing, laws are needed to completely describe electromagnetic field interaction with medium and source mechanisms. From [2],

\[
\int_S \mathbf{E} \cdot d\mathbf{S} = \frac{q}{\varepsilon_0} \quad \text{(2.1)}
\]

\[
\int_S \mathbf{B} \cdot d\mathbf{S} = 0 \quad \text{(2.2)}
\]

\[
\int_L \mathbf{E} \cdot d\mathbf{l} = -\int_S \frac{\partial \mathbf{B}}{\partial t} \cdot d\mathbf{S} \quad \text{(2.3)}
\]

\[
\int_L \mathbf{B} \cdot d\mathbf{l} = \mu_0(I + \varepsilon_0 \int_S \frac{\partial \mathbf{E}}{\partial t} \cdot d\mathbf{S}) \quad \text{(2.4)}
\]

Eq. (2.1) together with (2.2) are referred to as Gauss’ law or Gauss’ law for the electric and magnetic field, respectively. The significance of the first law is that the total electric

---

\(^1\)To fully constitute the basic framework of the electromagnetic theory the Lorentz force needs to be considered.
Table 1: The complete set of Maxwell’s equations

<table>
<thead>
<tr>
<th>Differential form</th>
<th>Integral form</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \nabla \times H = J + \frac{\partial D}{\partial t} )</td>
<td>( \int_L H \cdot dl = \int_S (J + \frac{\partial D}{\partial t}) \cdot dS )</td>
</tr>
<tr>
<td>( \nabla \times E = -\frac{\partial B}{\partial t} )</td>
<td>( \int_L E \cdot dl = -\int_S \frac{\partial B}{\partial t} \cdot dS )</td>
</tr>
<tr>
<td>( \nabla \cdot D = \rho_v )</td>
<td>( \int_S D \cdot dS = \int_v \rho_v , dv )</td>
</tr>
<tr>
<td>( \nabla \cdot B = 0 )</td>
<td>( \int_S B \cdot dS = 0 )</td>
</tr>
</tbody>
</table>

- **E** - Electric field intensity, \( [V/m] \)
- **D** - Electric flux density, \( [C/m^2] \)
- **\( \rho_v \)** - Volume charge density, \( [C/m^3] \)
- **\( \varepsilon \)** - Permittivity of the medium, \( [F/m] \)
- **H** - Magnetic field intensity, \( [A/m] \)
- **B** - Magnetic flux density, \( [Wb/m^2] \)
- **J** - Electric current density, \( [A/m^2] \)
- **\( \mu \)** - Permeability of the medium, \( [H/m] \)

\( D = \varepsilon E \)
\( B = \mu H \)
\( J = \sigma E \)

Flux through an arbitrary closed surface is proportional to the total net charge inside the surface. The second law states that the magnetic flux through a closed surface is always zero, meaning that there exist no magnetic monopoles. This is quite intuitive since the lines of force of the magnetic field are always closed.

The third of Maxwell’s equations, \( (2.3) \), is called the Faraday-Henry law and relates a time-varying magnetic field with an induced electromotive force. Eq. \( (2.4) \) is called the Ampère-Maxwell law. Ampère’s law relates the magnetic circulation to a current in a conductor and was modified in 1873 by Maxwell. The second term on the right hand side will be zero if the flux of the electric field does not vary with time, as in Ampère’s approach. The modification is a direct consequence of the principal of conservation of energy, and led Maxwell to his prediction of the existence of electromagnetic waves.

By using Gauss’ divergence theorem, Maxwell’s equations also take on a differential form. This form is desirable when dealing with volume integration. Table 1 shows the complete set of Maxwell’s equations and also includes three medium-dependent equations. Index 0 of \( \mu \) and \( \varepsilon \) of \( (2.1) \) and \( (2.4) \) indicates that the medium is vacuum. Throughout this thesis no magnetic medium will be considered, thus making \( \mu = \mu_0 \).
2.2 Introduction to PEEC

The PEEC method [3, 4, 5] arises from inductance calculations by Albert E. Ruehli at IBM T.J. Watson Research Center in 1970. The PEEC approach has been proven successful for the modelling of electromagnetic problems and allows for inclusion of lumped elements describing voltage sources, resistances, inductances and capacitors. This feature makes it, among other things, very suitable for modelling of printed circuit boards (PCBs) and since no discretization of the air is needed, large structures can be modelled without immense inherited computational complexity.

2.3 Derivation of basic PEEC theory

The scalar electrical potential, \( \phi \), is related to the electric field intensity according to (2.5). In analogy there exists a vector magnetic potential, \( A \), which is defined by (2.6).

\[
\mathbf{E} = -\nabla \phi \tag{2.5}
\]

\[
\nabla \times \mathbf{A} = \mathbf{B} \tag{2.6}
\]

If \( \nabla \cdot \mathbf{A} = 0 \) is imposed, the vector magnetic potential is given by [6]

\[
\mathbf{A} = \int_v \frac{\mu \mathbf{J}}{4\pi r_{\text{dist}}} \, dv, \tag{2.7}
\]

where \( \mathbf{J} \) is a current density vector and \( r_{\text{dist}} \) is the distance between the observer and the source. The potential from a charge distribution is given by [6]

\[
\phi = \int_v \frac{q}{4\pi \varepsilon r_{\text{dist}}} \, dv, \tag{2.8}
\]

where \( q \) is the charge density and \( r_{\text{dist}} \) is the distance between the observer and the charge distribution.

These expressions are presented under quasi-static conditions, meaning that the electromagnetic waves travel at an infinite speed. Since the waves do propagate at a finite speed, the expressions must be dependent on time. This is indicated below by introducing \( t \). The significance of \( r_{\text{dist}} \) in (2.7) and (2.8) stays the same, but since there is a strong need to handle more than one element, \( \mathbf{r} \) and \( \mathbf{r}_k' \) are introduced. The vector \( \mathbf{r} \) points from the origin to the observer, and each \( \mathbf{r}_k' \) is a vector pointing from the \( k \)-th source to the observer. The time delay is defined as \( |\mathbf{r} - \mathbf{r}_k'|/c \), where \( c \) is the speed of light in vacuum, or approximately \( 3 \cdot 10^8 \) m/s. In the frequency domain, the propagation delay results in a phase shift equal to \( \omega |\mathbf{r} - \mathbf{r}_k'|/c \).
2.3.1 Derivation of the electric field integral equation (EFIE)

The contents of the following subsections mainly originate from [7]. A basic understanding is important for realizing the complexity of large 3D structures, the limitations of the numerical models, and for the postprocessing explained in Chapter 4.

The starting point is to consider the total electric field, \( \mathbf{E}^T(r, t) \) in (2.9), to be the sum of a potential applied external electric field, \( \mathbf{E}^i(r, t) \), and a scattered field, \( \mathbf{E}^S(r, t) \).

\[
\mathbf{E}^T(r, t) = \mathbf{E}^i(r, t) + \mathbf{E}^S(r, t) \tag{2.9}
\]

The latter is considered to be the sum of the negative time derivative of the magnetic vector potential, and the negative gradient of the electric scalar potential, where

\[
\mathbf{E}^T(r, t) = \mathbf{E}^i(r, t) - \frac{\partial \mathbf{A}(r, t)}{\partial t} - \nabla \phi(r, t). \tag{2.10}
\]

The transformation from (2.10) to the EFIE is made by introducing the free space Green’s function (2.11) and the delayed time, \( t_d \), according to (2.12).

\[
G(r, r'_k) = \frac{1}{4\pi \frac{1}{|r - r'_k|}} \tag{2.11}
\]

\[
t_d = t - \frac{|r - r'_k|}{c} \tag{2.12}
\]

Under the condition that the observation point is on the surface of a conductor, the total electric field is equal to the ratio of the current density to the conduction, i.e.

\[
\mathbf{E}^T(r, t) = \frac{\mathbf{J}(r, t)}{\sigma}. \tag{2.13}
\]

Inserting (2.13), (2.7), (2.8), (2.11) and (2.12) in (2.10) finally yields

\[
\hat{n} \times \mathbf{E}^i(r, t) = \hat{n} \times \left[ \frac{\mathbf{J}(r, t)}{\sigma} \right] + \hat{n} \times \left[ \sum_{k=1}^{K} \mu \int_{v_k} G(r, r'_k) \frac{\partial \mathbf{J}(r'_k, t_d)}{\partial t} dv_k \right] + \hat{n} \times \left[ \sum_{k=1}^{K} \frac{\nabla}{\varepsilon_0} \int_{v_k} G(r, r'_k) q(r'_k, t_d) dv_k \right], \tag{2.14}
\]

where \( \hat{n} \) is the surface normal to the body surfaces. Note that the number of volume sources are indicated by \( K \) in (2.14).
2.3. Derivation of basic PEEC theory

2.3.2 Interpretation as equivalent circuit

To solve the integrals in (2.14), an approach that uses a concept of rectangular pulse functions is used. The structure is discretized into volume cells carrying a constant current density. The charges are supposed to be on the surfaces of the cells, and are also considered constant. The pulse functions are utilized in the PEEC models to mathematically describe this concept.

Figure 2.1 shows a 3D discretization of a rectangular conductor. The arrows indicate current direction, dashed lines separate volume cells while dotted lines separate surface cells. 1D and 2D discretization are also possible, the distinction will be the number of surface cells and the number of directions in which the currents exist. The use of 3D models should be avoided wherever possible, since the complexity significantly increases [8]. Figure 2.2 and 2.3 show a volume and a surface cell, both in one dimension, interpreted as equivalent circuits. The voltage sources can account for electromagnetic couplings from other cells.

2.3.3 Frequency domain circuit equations

The final frequency-domain PEEC model, which includes external sources and dielectric material, is shown in Figure 2.4. Applying Kirchhoff’s voltage and current laws to the branches and nodes of the equivalent circuit, results in the following equation system,

\[
\begin{bmatrix}
-C_M & -(R + j\omega L) \\
-j\omega F + S^T Y_L & -S^T C_M^T
\end{bmatrix}
\begin{bmatrix}
V \\
I_L
\end{bmatrix}
= \begin{bmatrix}
V_S \\
S^T I_S
\end{bmatrix},
\]

which is to be solved for the unknown potentials and currents, \(V\) and \(I_L\), respectively. \(R, L, F\) and \(S\) are matrices containing partial elements, \(C_M\) is defined for purpose of
facilitation, $Y$ is an admittance matrix describing the lumped components, while $V_S$ and $I_S$ are external sources [7].

The arrangement in (2.15) is called the modified nodal analysis (MNA) method and makes it possible to extract all currents and potentials of the PEEC structure. The nodal analysis (NA) method uses another approach to solve the equivalent circuits of Figure 2.4. The NA only solves for the node potentials and is hence faster [7].

In the implementation, the MNA method is used to extract the currents from all volume cells when calculating the electromagnetic field. The NA method is used when the input terminal of the antenna is probed. This is of course also possible using MNA, but not vice versa.

**Figure 2.2:** $(L_p)$PEEC model for a volume cell.

**Figure 2.3:** $(P)$PEEC model for a surface cell.
2.4 EISLAB’s PEEC solvers

In the spring of 2005 Frederik Schmid finished his Master’s thesis in computer science at EISLAB [8]. The result was a PEEC solver running in Linux. It was developed in C++ using the Portable, Extensible Toolkit for Scientific Computation (PETSc) [9], an open suite of data structures and routines for applications modelled by partial differential equations. The task was to optimize experimental code developed by researchers at EISLAB and the EMC Laboratory of the Dept. of Electrical Engineering at the University of L’Aquila. The previous work was code fragments solving various PEEC problems, lacking consistency and means of a generic input. There was also a desire to surrender a Windows implementation that used a commercial linear-algebra package.

During the fall of 2005 Peter Anttu has done a revised implementation of the PEEC solver. PETSc has been replaced by Gmm++ [10], which is a C++ template library for matrix models and operations. Gmm turned out to be more suitable for PEEC calculations, hence a more efficient solver is now available with a clear and intuitive source code. As of today, the work is focused on making a parallel implementation that is intended to run on a computer cluster at Umeå University.

Both PETSc-PEEC and Gmm-PEEC have been utilized during the work of this thesis. Without going into detail, the implementations consider text files as input, describing the physical properties of the model, the existence of external components and a number of parameters affecting how the model is to be treated by the solver. The output is a text file containing the desired potentials and currents formatted in Matlab syntax.

![Figure 2.4: \((L_p,R,P)\)PEEC model.](image-url)
3.1 Background

In 1995 James Kennedy and Russell Eberhart presented particle-swarm optimization (PSO), an optimizer that models the behavior and intelligence of a swarm of bees, school of fish or flock of birds, and emphasizes both social interaction and nostalgia from the individual’s perspective.

Kennedy and Eberhart formed a research duo of a social psychologist and an electrical engineer, whose work has had a great impact on the electromagnetic community. PSO is intuitive, easy to implement and has been proven to outperform other and more intricate methods like genetic algorithms.

3.2 Genetic algorithms

A basic understanding of genetic algorithms (GAs) is preferable when dealing with other optimization techniques, thus a short introduction will be given in this section. GAs were introduced in the early 70s by John Holland, and is sprung from evolutionary computing, invented in the 60s.

GAs rely on the principals of Darwin’s theory of evolution. In optimization applications the concept of survival of the fittest combined with selection and adaptation, provides robust and stochastic search methods. Being effective in optimizing complex,
multidimensional functions in a near-optimal fashion, GAs have proven successful in a vast amount of engineering problems.

Since GAs model evolution they, in essence, share paradigm with genetics. The key concepts are [11];

- **Genes** - A parameter is generally equivalent to a gene. Some sort of coding or mapping translates a parameter value into a gene,
- **Chromosomes** - A string of genes is referred to as a chromosome. In a 3D example three genes would together form a trail solution. This solution is equivalent to a chromosome or a position,
- **Population** - A set of solutions is called a population,
- **Generation** - Iterations in the GA,
- **Parent** - A pair of existing chromosomes are selected from the population for mating or recombination,
- **Child** - The offspring of the parents, and
- **Fitness** - There must be a way to tell how fit an individual is. Often, there exists a function which given a chromosome returns a fitness value.

The main idea is to represent parameters as genes in chromosomes. Valid chromosomes are grouped in a population, from which fit parents are selected to produce new chromosomes by recombination and mutation.

One reason for the fact that GAs are being considered complex and untidy to implement, is the many options associated with the selection, recombination and mutation [12]. The PSO, on the other hand, only concerns one major operation. This operation is the velocity calculation, which will be discussed in the following sections.

### 3.3 Theory

Since PSO models swarm behavior, this sections takes of from a somewhat informal point of view. Imagine a swarm of bees looking for the most fertile feeding location in a field. Each bee has a location in the three-dimensional space, $\mathbf{x}_m$, where the parameters $x_1$, $x_2$ and $x_3$ are intended to constitute a point in space. The bee evaluates every position for the absolute fitness. This fitness will, for this example, be a positive number which increases with increasing fertility. The bee remembers the spot where it encountered the best fitness and also shares this information with the other bees, so that the entire swarm will know the global best position. The bee’s movement is controlled by its velocity, $\mathbf{v}_m$, which is influenced by its best personally encountered location and the global best. The bee will always try to find the way back to its personal best location, while at the same
time curiously moving towards the global best. If a bee finds a location that has a fitness
greater than any encountered before, the entire swarm will be informed instantly and thus
move towards this location. The result will be a swarming behavior, evidently based on
both nostalgia and social influence.

It should be emphasized that the algorithm is considered to be continuous, i.e. the
particles’ parameters can take on any value in the defined interval. In the previous
example this means that the bee can be in any position in the field, even in the exact
same spot where other bees are. Another way of describing this is to state that collisions
do not occur, which of course lacks correspondence in real life.

A function that evaluates the position in solution space is needed. Though the algo-
rithm is generic, the fitness function is often unique to a specific problem.

More formally, the algorithm is [12];

- Define the solution space,
- Define a fitness function,
- Randomly initialize $x_m$ and $v_m$ (for particle 1 to $M$),
- Reckon $p_{bp}^m$ and $p_{bv}^m$ (for particle 1 to $M$),
- Reckon $g_{bp}$ and $g_{bv}$, and
- Until some criteria are met do (for particle 1 to $M$):
  - Evaluate current position’s fitness
    - If it is better than $p_{bv}^m$, exchange $p_{bp}^m$ and $p_{bv}^m$
    - If it is better than $g_{bv}$, exchange $g_{bp}$ and $g_{bv}$
  - Reckon $v_m$
  - Let $x_m = x_m + v_m$ (Determine next position),

where $p_{bp}^m$ is a vector pointing to the personal best position, and $p_{bv}^m$ is the value associated
with that position. $g_{bp}$ and $g_{bv}$ are in the same manner associated with the global best
position. Note that the personal position and value are related to one of the $M$ particles,
and that the global best position and value are shared, thus equal to all individuals. The
dimension is of course not limited to three, but instead $N$.

Of consistency, the reckoning of next position should read

$$x(t + \Delta t)_m = x(t)_m + v(t)_m \Delta t, \quad (3.1)$$

though $t$ is often omitted and $\Delta t$ is implied to be 1.

PSO is considered to be unique in the sense that it explores the space wherein the
solutions exist, not a space of solutions, which is a fundamental property of GAs.
3.3.1 Change of change

The heart of the optimization is the computation of the velocity, which is analogous to
the modification of the relative change, or simply change of change. The concept is to use
vectors pointing from the current to the personal best and global best position, according to

\[ v(t)_{mn} = v(t - \Delta t)_{mn} + \frac{\phi_1(p_{bp mn} - x_{mn}) + \phi_2(g_{bp n} - x_{mn})}{\Delta t}, \tag{3.2} \]

where

\[ \phi_1 = c_1 \text{rand()} \]

and

\[ \phi_2 = c_2 \text{rand()} \]

The constants \( c_1 \) and \( c_2 \) affect the influence of nostalgia and social interaction, respectively, whereas \( \text{rand()} \) is a function returning a random number from a uniform distribution in the interval \([0,1)\). The calls to the random function are considered to be separate, thus making them independent. Again, implying \( t \) and \( \Delta t = 1 \), a clearer representation of (3.2) is given by

\[ v_{mn} = v_{mn} + \phi_1(p_{bp mn} - x_{mn}) + \phi_2(g_{bp n} - x_{mn}). \]

Throughout the report, the significance of \( \phi_1 \), \( \phi_2 \) and \( \text{rand()} \) will not change.

3.3.2 Craziness, explosion, inertia and constraints

Since \( \phi_1 \) and \( \phi_2 \) are stochastic, their presence models the slight unpredictable behavior, or craziness, of particles in a swarm [13].

If the variables can take on any value, an oscillating behavior of increasing amplitude is likely to occur if the velocity is not constrained. This is often referred to as explosion and is avoided by limiting \( |v_m| \) by the positive number \( v_{\text{max}} \).

The variables are often limited to some interval. This will introduce a need to handle particles trying to pass the boundary limits. Figure 3.1 shows three boundary conditions. The leftmost shows the absorbing-wall approach, where the velocity in the direction of the boundary is zeroed. The following condition, the bouncing-wall approach, reflects the particles while the rightmost, the invisible wall, simply lets the particles pass. An inherited condition is that no fitness evaluation will be performed beyond the boundaries. This usually reduces the number of computations drastically, since the algorithm itself is very simple compared to most fitness evaluations. Particles outside the boundaries are supposed to, on their own, find their way back to the defined space [12].

Inertial weight, \( w \), is shown in (3.3) and was introduced for controlling the convergence of the algorithm. A large \( w \) encourages exploration, while a small \( w \) makes the particles fine-comb the area surrounding the global maximum. The inertial weight is therefor often
linearly decreased during an optimization, to speed up the convergence while covering a large area at the beginning.

\[ v_{mn} = w v_{mn} + \phi_1 (p_{mn}^{bp} - x_{mn}) + \phi_2 (g_{n}^{bp} - x_{mn}). \]  

(3.3)

The constriction factor, \( K \), was introduced to make an analytical analysis of the PSO, though (3.4) can be considered a special case of (3.3) [12].

\[ v_{mn} = K(v_{mn} + \varphi \text{rand}()(p_{mn}^{bp} - x_{mn}) + \varphi_2 \text{rand}()(g_{n}^{bp} - x_{mn})), \]  

(3.4)

where \( K \) is determined from

\[ \varphi = \varphi_1 + \varphi_2; \varphi > 4 \]  

(3.5)

and

\[ K = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}. \]  

(3.6)

Combinations of empirical testing and mathematical analysis of various test cases, by means of parameter settings, are summarized in [12]. The suggested settings are \( c_1 = 1.49, c_2 = 1.49 \), and that \( w \) is linearly decreased from 0.9 to 0.4, or that \( K = 0.729, \varphi_1 = 2.8 \text{ and } \varphi_2 = 1.3 \), depending on method of implementation.

Overall, a population size of \( \leq 30 \), using the invisible-wall approach with the settings displayed, has proven to provide good results [12].

![Figure 3.1: Boundary conditions in the PSO algorithm.](image)

### 3.3.3 Matrix representation

In [14], a matrix representation of the particle swarm is proposed. The \( M \) positions, velocities and personal best locations, all in \( N \) dimensions, are gathered in the \( M \times N \)
matrices, \(\mathbf{X}, \mathbf{V}\) and \(\mathbf{P}\), as described in (3.7), (3.8) and (3.9), respectively.

\[
\mathbf{X} = \begin{bmatrix}
x_{11} & x_{12} & \ldots & x_{1N} \\
x_{21} & x_{22} & \ldots & x_{2N} \\
\vdots \\
x_{M1} & x_{M2} & \ldots & x_{MN}
\end{bmatrix}
\]

(3.7)

\[
\mathbf{V} = \begin{bmatrix}
v_{11} & v_{12} & \ldots & v_{1N} \\
v_{21} & v_{22} & \ldots & v_{2N} \\
\vdots \\
v_{M1} & v_{M2} & \ldots & v_{MN}
\end{bmatrix}
\]

(3.8)

\[
\mathbf{P} = \begin{bmatrix}
p_{11} & p_{12} & \ldots & p_{1N} \\
p_{21} & p_{22} & \ldots & p_{2N} \\
\vdots \\
p_{M1} & p_{M2} & \ldots & p_{MN}
\end{bmatrix}
\]

(3.9)

The global best is still represented by a vector, but is referred to as a \(1 \times N\) matrix,

\[
\mathbf{G} = \begin{bmatrix}
g_1 \\
g_2 \\
\vdots \\
g_N
\end{bmatrix}
\]

The matrix representation leads to a very elegant expression for the new position of all particles,

\[
\mathbf{X} = \mathbf{X} + \mathbf{V}.
\]

The velocity matrix must still be updated element by element as,

\[
v_{mn} = v_{mn} + \phi_1(p_{mn} - x_{mn}) + \phi_2(g_n - x_{mn}),
\]

(3.10)

to retain the intended behavior.

Note that this is exactly what has been described in the previous section, though the value associated with each position is implied, and therefore the need for distinction between vector and scalar is no longer necessary.

### 3.4 Discrete binary representation

In [15], a clever technique for making the particle swarm and its operations binary is introduced. This implementation distinguishes the solution space from the coding space and uses a binary implementation of the position vectors, where \(\mathbf{x}_\mathbf{m} = [01 \ldots x_{mN}], \mathbf{p}_m = [00 \ldots p_{mN}]\) and \(\mathbf{g}_m = [10 \ldots g_N]\).

The velocity is given by

\[
v_{mn} = v_{mn} + \phi_1(p_{mn} - x_{mn}) + \phi_2(g_n - x_{mn}),
\]

(3.10)

where the velocity is not a binary number.
3.4. Discrete binary representation

3.4.1 Change of change of change

While the update of the velocity according to (3.10) stays the same, compared to that of the continuous version, the interpretation is now that velocity is the probability of a bit taking on a one or a zero. The meaning of the change of change is therefore evidently different. This is implemented as an IF statement,

\[ \text{IF (rand() < } S(v_{mn}) \text{) THEN } x_{mn} = 1 \text{ ELSE } x_{mn} = 0, \]

where

\[ S(v_{mn}) = \frac{1}{1 + e^{-v_{mn}}}, \tag{3.11} \]

is called a sigmoid limiting transformation [15].

This leads to the conclusion that the probability of a bit taking on a one is \( S(v_{mn}) \), and \( 1 - S(v_{mn}) \) that it will be a zero. Since this holds regardless of the initial state of a bit, the probability of a bit changing must be \( S(v_{mn})(1 - S(v_{mn})) \). Note that this only holds if the initial state of a bit is unknown. This is represented in [15] as

\[ p(\Delta) = S(v_{mn})(1 - S(v_{mn})). \tag{3.12} \]

Thus a change in the velocity still is a change in the rate of change [15].

3.4.2 Gray coding

Initial testing of function optimization showed that the particles often converged on a spot near the global best, and that they never tended to find the global best, no matter how many iterations that were performed. The cause is easily realized if the distance in coding space is considered.

In Table 1, the fitness of a solution space is illustrated. Table 2 shows two different representations of the coding space, related to the solution space in Table 1.

If the algorithm used the first representation in Table 2 and ended up on the position marked with bold figures, it would take a probability of all bits changing to end up on the global best, which of course is very low. In fact, it is a chance so slim that it can never be expected that the algorithm finds the global maximum. If, on the other hand, the second representation is used, only two bits need to change for the algorithm to find the global maximum.

To circumvent this problem, the desired mapping from coding to solution space is one that considers a short Hamming distance to represent a short distance in the solution space. One way of achieving this is to use a reflected binary Gray code shown in Table 3. The first implementation discussed uses a binary representation of increasing decimal numbers, while the second uses Gray code. The advantage is that adjacent positions will differ with a maximum Hamming distance of 2. One bit change does however not mean that an equivalent move of length one is executed, but it definitely makes the algorithm explore the space more freely.
Table 1: Fitness of the solution space.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.88</td>
<td>0.94</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>0.94</td>
<td>1.00</td>
<td>0.94</td>
</tr>
<tr>
<td>...</td>
<td>0.88</td>
<td>0.94</td>
<td>0.88</td>
</tr>
</tbody>
</table>

Table 2: Initial and Gray-code representation of coding space.

<table>
<thead>
<tr>
<th></th>
<th>0111110001₂</th>
<th>1000010001₂</th>
<th>1000110001₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>0111110000₂</td>
<td>1000010000₂</td>
<td>1000011000₂</td>
</tr>
<tr>
<td>...</td>
<td>0111101111₂</td>
<td>1000011111₂</td>
<td>1000101111₂</td>
</tr>
<tr>
<td>...</td>
<td>0100011001₂</td>
<td>1100011001₂</td>
<td>1100111001₂</td>
</tr>
<tr>
<td>...</td>
<td>0100001100₂</td>
<td>1100001100₂</td>
<td>1100101100₂</td>
</tr>
</tbody>
</table>

Table 3: Gray codes with corresponding decimal numbers.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00000₂</td>
<td>8</td>
<td>01100₂</td>
<td>16</td>
<td>11000₂</td>
<td>24</td>
<td>10100₂</td>
</tr>
<tr>
<td>1</td>
<td>00011₂</td>
<td>9</td>
<td>01101₂</td>
<td>17</td>
<td>11001₂</td>
<td>25</td>
<td>10110₂</td>
</tr>
<tr>
<td>2</td>
<td>00110₂</td>
<td>10</td>
<td>01111₂</td>
<td>18</td>
<td>11011₂</td>
<td>26</td>
<td>10111₂</td>
</tr>
<tr>
<td>3</td>
<td>00111₂</td>
<td>11</td>
<td>01110₂</td>
<td>19</td>
<td>11010₂</td>
<td>27</td>
<td>10110₂</td>
</tr>
<tr>
<td>4</td>
<td>00101₂</td>
<td>12</td>
<td>01010₂</td>
<td>20</td>
<td>11110₂</td>
<td>28</td>
<td>10010₂</td>
</tr>
<tr>
<td>5</td>
<td>00111₂</td>
<td>13</td>
<td>01011₂</td>
<td>21</td>
<td>11111₂</td>
<td>29</td>
<td>10011₂</td>
</tr>
<tr>
<td>6</td>
<td>00101₂</td>
<td>14</td>
<td>01001₂</td>
<td>22</td>
<td>11101₂</td>
<td>30</td>
<td>10001₂</td>
</tr>
<tr>
<td>7</td>
<td>00100₂</td>
<td>15</td>
<td>01000₂</td>
<td>23</td>
<td>11100₂</td>
<td>31</td>
<td>10000₂</td>
</tr>
</tbody>
</table>
When dealing with antennas, the radiated electromagnetic field is of great importance. Analytic evaluations are, however, often very cumbersome, even if extensive simplifications are made. In this chapter, a numerical method for calculating the radiated electromagnetic field from a PEEC structure is discussed.

4.1 The spherical coordinate system

In the orthogonal implementation of PEEC, all volume elements will be in the direction of either one of the base vectors in the cartesian coordinate system. If the volume cells are treated as short dipoles, generally known as Hertzian dipoles, convenient ways of calculating the electric field exist.

In Figure 4.1 a point in space, P, is represented. The vector pointing to P can be represented by \( \mathbf{r} = r_x \mathbf{i} + r_y \mathbf{j} + r_z \mathbf{k} \) or \( \mathbf{r} = r_R \mathbf{e}_R + r_\theta \mathbf{e}_\theta + r_\phi \mathbf{e}_\phi \). From the figure it is determined that \( r_x = r_R \sin \theta \cos \phi = R \sin \theta \cos \phi \), \( r_y = r_R \sin \theta \sin \phi \), \( r_z = r_R \cos \theta = R \cos \theta \). From this, expressions converting from spherical to cartesian coordinates can be gathered in a matrix, allowing for conversions according to

\[
\begin{bmatrix}
  a_x \\
  a_y \\
  a_z
\end{bmatrix} =
\begin{bmatrix}
  \sin \theta \cos \phi & \cos \theta \cos \phi & -\sin \phi \\
  \sin \theta \sin \phi & \cos \theta \sin \phi & \cos \phi \\
  \cos \theta & -\sin \theta & 0
\end{bmatrix}
\begin{bmatrix}
  a_R \\
  a_\theta \\
  a_\phi
\end{bmatrix},
\]

where \( \mathbf{a} \) is converted to a cartesian representation.
4.2 Electromagnetic field from a Hertzian dipole

In the far field of a Hertzian dipole the electric and magnetic field are completely in phase. It can be shown that they, for an infinitesimal current element in the direction of \( \mathbf{k} \), can be described by [6]

\[
E_\theta = Z_0 \frac{j I_z d\ell_z \beta}{4\pi r_{\text{dist}}} \sin \theta e^{-j\beta r_{\text{dist}}}
\]

and

\[
H_\phi = \frac{j I_z d\ell_z \beta}{4\pi r_{\text{dist}}} \sin \theta e^{-j\beta r_{\text{dist}}},
\]

where \( Z_0 = 120 \, \Omega \) is the free-space impedance, \( I_z \) the current, \( d\ell \) the infinitesimal length, and \( \beta = \omega/c = 2\pi f/c \) is the wave number. The distance between the observer and the center of the dipole is represented by \( r_{\text{dist}} \).

If the current is considered constant, the expressions for an infinitely thin dipole of finite length \( l_z \), placed at the origin and in the direction of \( \mathbf{k} \), will be

\[
E_\theta = Z_0 \frac{j I_z l_z \beta}{4\pi r_{\text{dist}}} \sin \theta e^{-j\beta r_{\text{dist}}}
\]

and

\[
H_\phi = \frac{j I_z l_z \beta}{4\pi r_{\text{dist}}} \sin \theta e^{-j\beta r_{\text{dist}}}.
\]

In similar fashion, the electric field radiating from a short dipole, placed in the direction of \( \mathbf{i} \), will be

\[
E_\phi = -Z_0 \frac{j I_x l_x \beta}{4\pi r_{\text{dist}}} \sin \phi e^{-j\beta r_{\text{dist}}}
\]
and if placed in the direction of $\mathbf{j}$, 
\[
E_\phi = -Z_0 \frac{jI_y \ell_y \beta}{4\pi r_{\text{dist}}} \cos \phi e^{-j\beta r_{\text{dist}}}. \tag{4.6}
\]

Letting the PEEC solver use the MNA method yields the current and potential of each volume and surface cell, respectively. By utilizing the currents and knowing the position, $r'_k$, and size of each volume cell, superpositioning of the electric field in a point, $r$, from $K$ volume cells is given by

\[
E(r) = \begin{bmatrix} E_R(r) \\ E_\theta(r) \\ E_\phi(r) \end{bmatrix} = \sum_{k=1}^{K} \begin{bmatrix} 0 \\ Z_0 \frac{jI_x \ell_x \beta}{4\pi r_k} \sin \theta e^{-j\beta r_{\text{dist}}} \\ -Z_0 \frac{jI_y \ell_y \beta}{4\pi r_k} \sin \phi e^{-j\beta r_{\text{dist}}} - Z_0 \frac{jI_z \ell_z \beta}{4\pi r_k} \cos \phi e^{-j\beta r_{\text{dist}}} \end{bmatrix} \tag{4.7}
\]

where $r_{\text{dist}}^k = |r'_k - r|$. $I_x^k$, $I_y^k$ and $I_z^k$ are currents in the direction of $i$, $j$ and $k$, respectively. Each volume cell holds a current vector

\[
I_k = \begin{bmatrix} I_x^k \\ I_y^k \\ I_z^k \end{bmatrix} \tag{4.8}
\]

in which only one element is non-zero. This makes it possible to use (4.7) to sum over all $K$ elements. The resultant field is meant to be converted to a representation in the cartesian coordinate system.

While the expression for $E_\theta$ holds for any point of observation, the expressions for $E_\phi$ are considered special cases for when the point of observation is on the plane of $ij$. Consistent evaluation of antennas is therefor limited to structures discretezed in the axis of $k$.

### 4.2.1 Extraction of currents

The Gmm-PEEC solver has been configured to return the information needed to calculate the electromagnetic field. The solver was modified to always include the min and max coordinate, and the direction of the current for each volume cell. In MNA mode, the vector containing $\mathbf{V}$ and $\mathbf{I}_L$ of (2.15) is returned, thus making the current for each source element available.

### 4.3 Antenna parameters

The radiated power and its pattern are important indications of an antenna’s properties. Some measures of antenna performance are defined in this section, which originate from
The power of an electromagnetic field is given by the time-average of the Poynting vector
\[
\mathcal{P}_{\text{avg}} = \Re(\mathbf{E} \times \mathbf{H}^*) ,
\]
where the \( \mathbf{E} \) and \( \mathbf{H} \) fields are considered to be represented by their effective or RMS value.

The radiation intensity \( U(\theta, \phi) \) is defined as time-averaged power per unit solid angle as
\[
U(\theta, \phi) \equiv \frac{dP_{\text{rad}}}{d\Omega} = r^2 |\mathcal{P}_{\text{avg}}| = r^2 |\Re(\mathbf{E} \times \mathbf{H}^*)| = \frac{r^2}{Z_0} |\mathbf{E}|^2 .
\]

The directive gain \( D_{\text{gain}}(\theta, \phi) \) of an antenna is defined as the ratio of the radiation intensity to that of an isotropic radiator. An isotropic radiator is a hypothetical antenna that radiates the same total power at any point on a hypothetical sphere surrounding it as
\[
U_0 = \frac{P_{\text{rad}}}{4\pi} ,
\]
yielding
\[
D_{\text{gain}}(\theta, \phi) = \frac{U(\theta, \phi)}{U_0} = \frac{4\pi U(\theta, \phi)}{P_{\text{rad}}} .
\]

If the radiated power is considered to be equal to the power received by the antenna, the directive gain is given by
\[
D_{\text{gain}}(\theta, \phi) = \frac{4\pi U(\theta, \phi)}{P_{\text{in}}} = \frac{4\pi U(\theta, \phi)}{\Re(\mathbf{V}_{\text{in}} \mathbf{I}_{\text{in}}^*)} ,
\]
where \( \mathbf{V}_{\text{in}} \) and \( \mathbf{I}_{\text{in}} \) are complex phasors of the voltage and current at the feeding point. The asterisk, \( * \), indicates that the current is represented by its complex conjugate. Directivity of an antenna is considered to be the maximum value of its directive gain. The gain is often expressed in decibels, where
\[
G_{\text{dB}}(\theta, \phi) = 10 \log_{10} D_{\text{gain}}(\theta, \phi) ,
\]
still under the assumption that \( P_{\text{rad}} = P_{\text{in}} \).

Another factor considered is the return loss, \( S_{11} \). It is defined as the ratio of the reflected wave to that of the outgoing, experienced by the generator supplying the antenna. The return loss can be evaluated considering the impedances of the transmission line, connecting the generator and the antenna, and the impedance of the feeding point, where
\[
S_{11} = \frac{Z_{\text{antenna}} - Z_{\text{coax}}}{Z_{\text{antenna}} + Z_{\text{coax}}} .
\]

Common impedances of coaxial cables, used as transmission lines, are 50 and 75 \( \Omega \). It is desired to keep \( S_{11} \) as close to 0 as possible, thus matching the antenna with the feed for minimum power loss.
4.4 Verification of method

The PEEC solver was configured to model a long dipole, consisting of two rectangular bars. The bars are each 10 cm long, with a cross-sectional area of $10^{-10}$ cm$^2$. A 1A sinusoidal source at the origin connects the bars which expand on the axis of $i$, in opposite direction.

Each bar is discretized 50 times into 50 volume cells, in the direction of $i$, and the antenna is fed at its theoretical resonant frequency of 750 MHz. The absolute value of the electric field at a distance of 3 m from the origin is observed. Figure 4.2 shows the absolute value of the electric field from the PEEC structure compared to that of a theoretical long dipole. The expressions for the theoretical long dipole originate from [17], where the theoretical intensity is determined to 20 V/m, for this particular example.

The result from the superpositioning of the electric field, radiating from the 100 short dipoles in the PEEC structure, was a maximum field intensity of 20.082 V/m, which implies a deviation of 0.4 %.

![Electric field from a long dipole](image)

*Figure 4.2: Electric field from a long dipole.*
This chapter presents an implementation of the PSO algorithm optimizing both known and unknown functions. The PSO algorithm has been implemented in Matlab. The electromagnetic structures are modelled using PETSc-PEEC and Gmm-PEEC, and the result is postprocessed in Matlab. The algorithm is implemented in its binary guise and the antennas being evaluated are a binary patch and a dipole array.

5.1 Building blocks

The heart of the implementation is PSO.m which incorporates a binary version of the PSO. When a particle is to be evaluated for fitness a binary vector is passed to Binary-Patch.m.

BinaryPatch.m takes two additional arguments, telling whether the optimization should be visualized and if the PEEC solver should use the MNA or the NA method. Binary-Patch.m creates an input file to the PEEC solver, runs the external solver and evaluates the fitness, which is returned to the calling function. If the optimization is to be visual, the binary vector is passed to MyFunction.m which returns a fitness for a mathematical function as shown in Figure 5.1.

In the case of electromagnetic optimization, shown in Figure 5.2, BinaryPatch.m evaluates the impedance of the structure or the electromagnetic field. The latter is done by a call to EvalAntenna.m which returns the maximum directive gain, found in two horizontal planes, and the return loss. EvalAntenna.m calls Field.m, which implements the methods described in Chapter 4.

The electromagnetic problems are represented by binary patches, of which the PSO algorithm can alter the physical representations. Figure 5.3 shows the patch represented by \( \mathbf{b} = [1000011000110011] \). The lower left corner is represented by the first entry of \( \mathbf{b} \). The next position to the right, is the next entry. The positions are incremented row-wise,
up to the top right corner, represented by the last entry of $b$.

Appendix A contains the Matlab code of the functions mentioned in this section.

---

**Figure 5.1: Function optimization.**

**Figure 5.2: Antenna optimization.**
5.2 Optimization of a known function

To visualize the activities of the algorithm the optimizer was set to find the maximum of a polar sinc function,

\[ r_z = \frac{\sin(\sqrt{r_x^2 + r_y^2})}{\sqrt{r_x^2 + r_y^2}}, \quad (5.1) \]

shown in figure 5.4. The length of the binary vector is set to ten and two five-bit numbers are used to represent a position, where

\[ b = [11000_2, 11000_2]. \]

Figure 5.3: Example of a 4-by-4 patch antenna. The feeding point is indicated by the ring.

Figure 5.4: Radial sinc function.
The numbers in \( b \) are interpreted as Gray code, as discussed in Chapter 3. Figure 5.5, 5.6 and 5.7 show the progress of the optimization, and that the algorithm eventually finds the global maximum of the function. If two or more particles end up in the same spot, their individual fitness values are added, indicating that there exist nine particles at the global maximum in figure 5.7. Note that the figure is scaled to show the particles’ added fitness.

\[
\frac{\text{abs}(\sin(\sqrt{r_x^2+r_y^2})))}{\sqrt{r_x^2+r_y^2}}
\]

Figure 5.5: Particle locations after one iteration.
5.2. Optimization of a known function

Figure 5.6: Particle locations after ten iterations.

Figure 5.7: Particle locations after 39 iterations.
Results

6.1 Antenna optimization

Antennas are very untidy to handle by means of mathematical analysis. Instead, the antennas will be treated as composed of many infinitesimal dipoles, as previously discussed. This section considers a patch antenna that is placed at the origin in the $ij$-plane. The PSO controls the input to the PEEC solver and can hence alter the physical description of the antenna structures. The patch antenna is fed by a 1A current source in parallel with a 50 $\Omega$ resistor, representing a coaxial cable. The target for the PSO is to optimize resonant frequency.

The other configuration considered is an array of two long dipoles, with variable distance between the dipoles. Here, the electric-field strength is evaluated, and the directive gain is optimized.

6.1.1 Optimization of resonant frequency

As a first example, a resonant-frequency optimization of a binary patch antenna is performed. The target is a full binary patch of size 4 by 4 cm fed at (1,1), as shown in figure 6.1, together with the 1 A current source and the 50 $\Omega$ resistor. The absolute value of the current (bottom) flowing into the antenna, and the potential (top) of the feeding point are shown in Figure 6.2. The current and potential are assumed to have zero phase, i.e. be real-valued, at the resonant frequencies.

The fitness function was set to count the number of resonant frequencies, by examining the derivative of $|I_{in}|$. A structure not having exactly two resonant frequencies, as the target, was given a fitness of 0. All other structures are given a fitness equal to 1 minus the deviation of each resonant frequency from the target’s of 1.9 and 4 GHz. Any negative
fitness is set to 0. The target is considered an invalid configuration, as is any configuration not having a PEEC cell adjacent to the feeding point.

![Diagram of patch antenna](image)

*Figure 6.1: Model setup of the patch antenna.*

![Graph of voltage and current](image)

*Figure 6.2: Input current (bottom) and input voltage (top) at the feed of the reference antenna.*

The optimization was run with 10 particles and was stopped after 40 iteration. Table
1 and 2 show extracts from the log file. The rows in $\mathbf{X}$ hold current position of the particles while the rows in $\mathbf{P}$ show the personal best, according to Section 3.3.3. The left and right fitness column show current and personal best fitness, respectively. Line one of each extract shows which particle whose personal best is constituting the global best.

The antenna $\mathbf{b} = [010111000101010]$, was obviously found to be most fit and its characteristics are shown in Figure 6.3. It is evident that the optimized antenna has the desired resonant frequencies at 1.9 and 4 GHz. Since this antenna is physically smaller than the reference, this optimization could also be considered aiming at low weight.

**Table 1: The first iteration of the resonant-frequency optimization.**

<table>
<thead>
<tr>
<th>Iteration: 1 Global best: Particle 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbf{X}(1,:)$: 100000001101000 Fitness:0.636 $\mathbf{P}(1,:)$: 100000001101000 Fitness:0.636</td>
</tr>
<tr>
<td>$\mathbf{X}(2,:)$: 1011110010000011 Fitness:0.000 $\mathbf{P}(2,:)$: 1011110010000011 Fitness:0.000</td>
</tr>
<tr>
<td>$\mathbf{X}(3,:)$: 0001110001111110 Fitness:0.768 $\mathbf{P}(3,:)$: 0001110001111110 Fitness:0.768</td>
</tr>
<tr>
<td>$\mathbf{X}(4,:)$: 0111110100111010 Fitness:0.000 $\mathbf{P}(4,:)$: 0111110100111010 Fitness:0.000</td>
</tr>
<tr>
<td>$\mathbf{X}(5,:)$: 0000111001111110 Fitness:0.000 $\mathbf{P}(5,:)$: 0000111001111110 Fitness:0.000</td>
</tr>
<tr>
<td>$\mathbf{X}(6,:)$: 0101110101010110 Fitness:0.000 $\mathbf{P}(6,:)$: 0101110101010110 Fitness:0.000</td>
</tr>
<tr>
<td>$\mathbf{X}(7,:)$: 0100000010011011 Fitness:0.000 $\mathbf{P}(7,:)$: 0100000010011011 Fitness:0.000</td>
</tr>
<tr>
<td>$\mathbf{X}(8,:)$: 0111110111011011 Fitness:0.972 $\mathbf{P}(8,:)$: 0111110111011011 Fitness:0.972</td>
</tr>
<tr>
<td>$\mathbf{X}(9,:)$: 0100100000010110 Fitness:0.000 $\mathbf{P}(9,:)$: 0100100000010110 Fitness:0.000</td>
</tr>
<tr>
<td>$\mathbf{X}(10,:)$: 0010011101011000 Fitness:0.000 $\mathbf{P}(10,:)$: 0010011101011000 Fitness:0.000</td>
</tr>
</tbody>
</table>

**Table 2: The last iteration of the resonant-frequency optimization.**

<table>
<thead>
<tr>
<th>Iteration: 40 Global best: Particle 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbf{X}(1,:)$: 0111111010001110 Fitness:0.000 $\mathbf{P}(1,:)$: 01111111001110 Fitness:0.908</td>
</tr>
<tr>
<td>$\mathbf{X}(2,:)$: 0101001011100100 Fitness:0.672 $\mathbf{P}(2,:)$: 0101100101000010 Fitness:0.980</td>
</tr>
<tr>
<td>$\mathbf{X}(3,:)$: 0101111001001010 Fitness:0.992 $\mathbf{P}(3,:)$: 0101111001001010 Fitness:0.992</td>
</tr>
<tr>
<td>$\mathbf{X}(4,:)$: 0011111001001010 Fitness:0.804 $\mathbf{P}(4,:)$: 0101110100101010 Fitness:0.992</td>
</tr>
<tr>
<td>$\mathbf{X}(5,:)$: 0101110000101010 Fitness:0.940 $\mathbf{P}(5,:)$: 010111000101010 Fitness:0.992</td>
</tr>
<tr>
<td>$\mathbf{X}(6,:)$: 0101010001010111 Fitness:0.796 $\mathbf{P}(6,:)$: 010101001111001 Fitness:0.984</td>
</tr>
<tr>
<td>$\mathbf{X}(7,:)$: 0101111001010110 Fitness:0.992 $\mathbf{P}(7,:)$: 0101111001010110 Fitness:0.992</td>
</tr>
<tr>
<td>$\mathbf{X}(8,:)$: 0101111010101010 Fitness:0.000 $\mathbf{P}(8,:)$: 010111100101010 Fitness:0.992</td>
</tr>
<tr>
<td>$\mathbf{X}(9,:)$: 0001101100101010 Fitness:0.000 $\mathbf{P}(9,:)$: 0001101100101010 Fitness:0.976</td>
</tr>
<tr>
<td>$\mathbf{X}(10,:)$: 0111111010101010 Fitness:0.000 $\mathbf{P}(10,:)$: 0101111000101010 Fitness:0.992</td>
</tr>
</tbody>
</table>
6.1.2 Optimization of the electric-field strength

The patch antennas turned out to be complicated to model, when it comes to field optimization. However, since the dipole was proven to be modelled with good results, an array with such elements replaces the patch antenna for this purpose. BinPatch.m has been replaced by BinDipole.m, but works conceptually in the same way. Figure 6.4 shows the array used for the electric-field optimization, where \( a \) represents the length of the dipoles and \( b \) half the distance between them. The total distance, or \( 2b \), is labelled \( d \).

The directivity is evaluated in the planes of \( jk \) and \( ij \). The variable \( b \) is controlled by the PSO, using a five-bit binary number, interpreted as Gray code. The distance is varied between 0 and \( \lambda \) in 32 steps, while the directivity is being optimized. Table 3 and 4 show extracts from the log file. When the run was terminated after 10 iterations, the best fitness had been discovered by particle one to nine. 01111\(_2\) interpreted as Gray code.
6.1. Antenna optimization

Figure 6.4: Model setup of the dipole array.

Table 3: The first iteration of the directivity optimization.

<table>
<thead>
<tr>
<th>Iteration: 1</th>
<th>Global best: Particle 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>X(1,:) 00110</td>
<td>P(1,:) 00110 Fitness:3.290</td>
</tr>
<tr>
<td>X(2,:) 00101</td>
<td>P(2,:) 00101 Fitness:4.639</td>
</tr>
<tr>
<td>X(3,:) 01110</td>
<td>P(3,:) 01110 Fitness:6.853</td>
</tr>
<tr>
<td>X(4,:) 01110</td>
<td>P(4,:) 01110 Fitness:6.853</td>
</tr>
<tr>
<td>X(5,:) 01100</td>
<td>P(5,:) 01100 Fitness:6.128</td>
</tr>
<tr>
<td>X(6,:) 01001</td>
<td>P(6,:) 01001 Fitness:5.629</td>
</tr>
<tr>
<td>X(7,:) 01101</td>
<td>P(7,:) 01101 Fitness:6.662</td>
</tr>
<tr>
<td>X(8,:) 00000</td>
<td>P(8,:) 00000 Fitness:2.120</td>
</tr>
<tr>
<td>X(9,:) 11100</td>
<td>P(9,:) 11100 Fitness:5.236</td>
</tr>
<tr>
<td>X(10,:) 11011</td>
<td>P(10,:) 11011 Fitness:3.877</td>
</tr>
</tbody>
</table>

represents 10, which implies that the distance is $b = 10\lambda/31 \Rightarrow d = 20\lambda/31$. This result agrees well with the theoretical two-dipole array [18], where the maximum directive gain is found when $d = 2\lambda/3$. If $d$ is manually configured to $2\lambda/3$, the directivity found from the array modelled by the PEEC solver is 4.68 or 6.87 dB. Figure 6.5 shows the electric field intensity at a distance of three meters from the array. The figure is showing the plane of $ij$, in which the maximum directive gain was found.
Table 4: The last iteration of the directivity optimization.

<table>
<thead>
<tr>
<th>X(i,:)</th>
<th>Fitness</th>
<th>P(i,:)</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>00111</td>
<td>3.914</td>
<td>01111</td>
<td>6.868</td>
</tr>
<tr>
<td>01111</td>
<td>6.868</td>
<td>01111</td>
<td>6.868</td>
</tr>
<tr>
<td>01111</td>
<td>6.868</td>
<td>01111</td>
<td>6.868</td>
</tr>
<tr>
<td>01111</td>
<td>6.868</td>
<td>01111</td>
<td>6.868</td>
</tr>
<tr>
<td>01111</td>
<td>6.868</td>
<td>01111</td>
<td>6.868</td>
</tr>
<tr>
<td>01111</td>
<td>6.868</td>
<td>01111</td>
<td>6.868</td>
</tr>
<tr>
<td>01111</td>
<td>6.868</td>
<td>01111</td>
<td>6.868</td>
</tr>
<tr>
<td>01011</td>
<td>6.311</td>
<td>01011</td>
<td>6.311</td>
</tr>
<tr>
<td>01011</td>
<td>6.311</td>
<td>01011</td>
<td>6.311</td>
</tr>
</tbody>
</table>

Figure 6.5: Electric field from the dipole array.
Chapter 7

Conclusions and further work

7.1 Conclusions

The thesis presents how to combine the PSO algorithm with a PEEC-based electromagnetic solver, for the purpose of optimizing antenna structures. The intuitive behavior of the PSO algorithm was easily implemented and the result was overall very good. The merging of the existing PEEC solvers and the particle-swarm optimization algorithm turned out to be successful. The results indicate that the optimization is not limited to antennas, but could also be used for inverse problems and the design of micro-strip filters.

The field calculation turned out to be very tedious, since the extraction of currents never were intended while the structure of the object-oriented code was developed. Due to the definition of the spherical coordinate system, the results are only valid if the structures are 1-dimensional in the plane of $ik$ or $jk$, or if the point of observation lies in the plane of $ij$ for 3D structures. This is something that should be taken care of before using the resulting code as a generic optimizer.

The behavior was stable and the activities of the algorithm can be viewed and backtracked, by the use of a log.

It should be noted that the attached Matlab code is to be considered a draft, though it hopefully will be an aid for others interested in implementing the algorithm.

7.2 Further work

The first improvement would be to endow the PSO algorithm with a memory for the fitness of a certain antenna configuration. Since the structures being used in this thesis are relatively small, the penalty for evaluating the same configuration multiple times does
not degrade the overall performance significantly. For larger structures, it would however not be acceptable to reevaluate multi-hour runs.

It would be interesting to include an implementation of the genetic algorithm and thus be able to compare their progress. An elaborate investigation of how the PSO parameters affect this particular antenna optimization is also of interest.

The field evaluation of the dipole array was used mainly to show that the field optimization works. The initial idea of optimizing patch antennas did not turn out as expected, as was therefore partly abandoned. Since the patch antennas are important in the current research projects at EISLAB, the work involving modelling of these antennas will continue.

I have been advised to change the core of the binary version of the PSO. This proposal would make the movement equivalent to that of the continuous version and would preserve the swarming behavior of the particles. The concept is to use an alternate version of the presented Gray code and has, to my knowledge, never been implemented.
APPENDIX A

Matlab code

PSO.m:

```matlab
0001 clear all;
0002 if(length(strfind(path,'/mypeec/common'))==0)
0003    path(path,'/mypeec/common');
0004 end
0005 RAND('state',sum(100*clock));%Do not use pseudo-random numbers
0006 numOfParticles=10;
0007 numOfIterations=40;
0008 sizeOfArray=16;
0009 fid0=fopen('log.txt','w');
0010 visual=0;
0011 if visual==1
0012    [x,y]=meshgrid([-10:0.625:9.75],[-10:0.625:9.75]);
0013    r=sqrt(x.^2+y.^2)+eps;
0014    z=abs(sin(r))./r; %Radial sinc function
0015 end
0016 Vmax=6;
0017 w=0.729;
0018 c1=1.494;c2=1.494;
0019 mode='MNA';
0020 X=zeros(numOfParticles,sizeOfArray+1);
0021 for particle=1:numOfParticles %Check for invalid configurations
0022    if X(particle,1)==0&&X(particle,2)==0&&
0023       X(particle,5)==0&&X(particle,6)==0
0024       X(particle,1)=1;
0025    elseif X(particle,1:sizeOfArray)==ones(1,sizeOfArray)
0026       X(particle,1)=0;
0027 end
0028 end
0029 end
0030 Xnext=zeros(size(X));
0031 tempBest=zeros(numOfParticles,1);
0032 P=rand(numOfParticles,sizeOfArray)*Vmax*2-Vmax;
0033 P(:,1)=X;
0034 for particle=1:numOfParticles %Let all particles have
0035    P(particle,1:sizeOfArray)=round(rand(numOfParticles,sizeOfArray));
0036 end
0037 for iteration=1:numOfIterations
0038    fprintf(fid0,'Iteration: %i Global best: Particle %i
',...%
0039    if visual==1
0040        P(:,sizeOfArray+1)=find(P(:,sizeOfArray+1)==max(P(:,sizeOfArray+1))).'
0041    else
0042        P(:,sizeOfArray+1)=find(P(:,sizeOfArray+1)==max(P(:,sizeOfArray+1))).'
0043        BinPatch(P(:,1:sizeOfArray),-mode);
0044 end
0045 end
0046 end
```

APPENDIX A

Matlab code

PSO.m:

```matlab
0001 clear all;
0002 if(length(strfind(path,'/mypeec/common'))==0)
0003    path(path,'/mypeec/common');
0004 end
0005 RAND('state',sum(100*clock));%Do not use pseudo-random numbers
0006 numOfParticles=10;
0007 numOfIterations=40;
0008 sizeOfArray=16;
0009 fid0=fopen('log.txt','w');
0010 visual=0;
0011 if visual==1
0012    [x,y]=meshgrid([-10:0.625:9.75],[-10:0.625:9.75]);
0013    r=sqrt(x.^2+y.^2)+eps;
0014    z=abs(sin(r))./r; %Radial sinc function
0015 end
0016 Vmax=6;
0017 w=0.729;
0018 c1=1.494;c2=1.494;
0019 mode='MNA';
0020 X=zeros(numOfParticles,sizeOfArray+1);
0021 for particle=1:numOfParticles %Check for invalid configurations
0022    if X(particle,1)==0&&X(particle,2)==0&&
0023       X(particle,5)==0&&X(particle,6)==0
0024       X(particle,1)=1;
0025    elseif X(particle,1:sizeOfArray)==ones(1,sizeOfArray)
0026       X(particle,1)=0;
0027 end
0028 end
0029 end
0030 Xnext=zeros(size(X));
0031 tempBest=zeros(numOfParticles,1);
0032 P=rand(numOfParticles,sizeOfArray)*Vmax*2-Vmax;
0033 P(:,1)=X;
0034 for particle=1:numOfParticles %Let all particles have
0035    P(particle,1:sizeOfArray)=round(rand(numOfParticles,sizeOfArray));
0036 end
0037 for iteration=1:numOfIterations
0038    fprintf(fid0,'Iteration: %i Global best: Particle %i
',...%
0039    if visual==1
0040        P(:,sizeOfArray+1)=find(P(:,sizeOfArray+1)==max(P(:,sizeOfArray+1))).'
0041    else
0042        P(:,sizeOfArray+1)=find(P(:,sizeOfArray+1)==max(P(:,sizeOfArray+1))).'
0043        BinPatch(P(:,1:sizeOfArray),-mode);
0044 end
0045 end
0046 end
0047 end
0048 end
0049 end
0050 X(:,sizeOfArray+1)=P(:,sizeOfArray+1);%Since the current fitness is
0051 Exprited to the log, a value is
0052 Needed.
0053 %End of initialization
0054 for iteration=1:numOfIterations
0055    fprintf(fid0,'Iteration: %i Global best: Particle %i\n',...%
0056```
% Print to the log
for particle=1:numOfParticles
    if particle<10
        fprintf(fid0,'X(%i,:): ',particle);
    else
        fprintf(fid0,'X(%i,:):',particle);
    end
    for n=1:sizeOfArray
        if X(particle,n)==1
            fprintf(fid0,'1');
        else
            fprintf(fid0,'0');
        end
    end
    if particle<10
        fprintf(fid0,' Fitness:%.3f P(%i,:): ',X(particle,sizeOfArray+1),particle);
    else
        fprintf(fid0,' Fitness:%.3f P(%i,:):',X(particle,sizeOfArray+1),particle);
    end
    for n=1:sizeOfArray
        if P(particle,n)==1
            fprintf(fid0,'1');
        else
            fprintf(fid0,'0');
        end
    end
    fprintf(fid0,' Fitness:%.3f',P(particle,sizeOfArray+1));
end
for particle=1:numOfParticles
    % Recon vmn
    VTemp=w*V(particle,:)+rand()*(P(particle,1:sizeOfArray)-X(particle,1:sizeOfArray))+rand()*(P(gBest,1:sizeOfArray)-X(particle,1:sizeOfArray));
    if VTemp>Vmax
        V(particle,:)=Vmax;
    elseif VTemp<-Vmax
        V(particle,:)=-Vmax;
    else
        V(particle,:)=VTemp;
    end
end
for particle=1:numOfParticles
    for n=1:sizeOfArray
        if rand()<1/(1+exp(-V(particle,n)))% Logistic function
            Xnext(particle,n)=1;
        else
            Xnext(particle,n)=0;
        end
    end
end
for particle=1:numOfParticles
    % Check for invalid configurations
    if Xnext(particle,1)==0&&Xnext(particle,2)==0&&
        Xnext(particle,5)==0&&Xnext(particle,6)==0
        Xnext(particle,1)=1;
    elseif Xnext(particle,1:sizeOfArray)==ones(1,sizeOfArray)
        Xnext(particle,1)=0;
    end
end
for particle=1:numOfParticles
    if visual==1
        tempBest = BinPatch(Xnext(particle,1:sizeOfArray), 'visual');
    else
        tempBest = BinPatch(Xnext(particle,1:sizeOfArray), 'antenna', mode);
    end
    if tempBest > P(particle,1:sizeOfArray)
        P(particle,1:sizeOfArray)=tempBest;
    end
    if tempBest > P(gBest,1:sizeOfArray)
        gBest = particle;
    end
    Xnext(particle,sizeOfArray+1)=tempBest; % Save current fitness in the log
end
X=Xnext;
% Start of visualization section
if visual==1
    particleLoc=NaN(size(y));
    for particle=1:numOfParticles
        xbin=[]; ybin=[];
        for count=1:sizeOfArray/2
            if X(particle,count)==1
41

0141 xbin=\texttt{[xbin '1']};
0142 \textbf{else}
0143 xbin=\texttt{[xbin '0']};
0144 \textbf{end}
0145 \textbf{end}
0146 xpos=\texttt{gray2dec}(xbin);
0147 for \texttt{count}=\texttt{sizeOfArray}/2+1:\texttt{sizeOfArray}
0148 \textbf{if} \texttt{X(particle,count)}==1
0149 ybin=\texttt{[ybin '1']};
0150 \textbf{else}
0151 ybin=\texttt{[ybin '0']};
0152 \textbf{end}
0153 \textbf{end}
0154 ypos=\texttt{gray2dec}(ybin);
0155 \textbf{if} \texttt{(isnan(particleLoc(ypos+1,xpos+1)))}
0156 particleLoc(ypos+1,xpos+1)=\texttt{z(ypos+1,xpos+1)};
0157 \textbf{else}
0158 particleLoc(ypos+1,xpos+1)=\texttt{...}
0159 particleLoc(ypos+1,xpos+1)+\texttt{z(ypos+1,xpos+1)};
0160 \textbf{end}
0161 \textbf{end}
0162 contour3(\texttt{x,y,z});\texttt{hold on};
0163 xlabel(\texttt{'Parameter 1 (r_x)'})
0164 ylabel(\texttt{'Parameter 2 (r_y)'})
0165 zlabel(\texttt{'abs(sin(sqrt(r_x^2+r_y^2)))/sqrt(r_x^2+r_y^2)'})
0166 stem3(\texttt{x,y,particleLoc});\texttt{hold off};
0167 pause(0.1);
0168 \textbf{end}
0169 fclose(\texttt{fid0});\texttt{fclose log}

\textbf{BinPatch.m:}

0001 \texttt{function fitness = BinPatch(varargin)}
0002 \texttt{b=varargin{1};}
0003 \texttt{MNA=0;}
0004 \texttt{if} \texttt{length(varargin)>1&&strcmp(upper(varargin(1,2))},\texttt{'VISUAL'})
0005 \texttt{fitness=MyFunction(b);}
0006 \texttt{return}
0007 \texttt{end}
0008 \texttt{if} \texttt{length(varargin)>2&&strcmp(upper(varargin(1,3))},\texttt{'MNA'})
0009 \texttt{MNA=1;}
0010 \texttt{end}
0011 \texttt{if} \texttt{b(1)==0\&\&b(2)==0\&\&b(5)==0\&\&b(6)==0}
0012 \texttt{error('Invalid location of current source')}
0013 \texttt{end}
0014 \texttt{if} \texttt{b==\texttt{ones}(1,16)}\texttt{This is the reference}
0015 \texttt{error('Reference antenna found')}
0016 \texttt{end}
0017 \texttt{fid0=fopen('Geometry.inp','w');}
0018 \texttt{fprintf(fid0,'number of bars ' num2str(sum(b)) 'n');}
0019 \texttt{for} \texttt{count}=1:sum(b)
0020 \texttt{fprintf(fid0,'ndiva 2 ndivb 2 ndivc 0 thinthickness yes
');}
0021 \texttt{fprintf(fid0,\texttt{%.cs sigma 574e6
});}
0022 \texttt{fprintf(fid0,\texttt{%.bz 0 0 0 1 0 0 0 1 0 1 1 1 1 1 9
});}
0023 \texttt{for} \texttt{y = 0:sqrt(length(b))-1}
0024 \texttt{for} \texttt{x = 0:sqrt(length(b))-1}
0025 \texttt{if} \texttt{b((y+1)*sqrt(length(b))+x)==1}
0026 \texttt{p1=p1+\texttt{num2str(squareSize)} '}\texttt{'};
0027 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0028 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0029 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0030 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0031 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0032 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0033 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0034 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0035 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0036 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0037 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0038 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0039 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0040 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0041 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0042 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0043 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0044 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0045 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0046 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0047 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0048 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0049 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0050 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0051 \texttt{p2=p2+\texttt{num2str(squareSize)} '}\texttt{'};
0052 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};
0053 \texttt{num2str(squareSize)} \texttt{'}\texttt{'};

num2str((x+1)*squareSize) ' ' num2str((x+1)*squareSize)
num2str((z+thickness)*squareSize) ' ' num2str((z+thickness)*squareSize)

fprintf(fid0,['.cs sigma ' num2str(sigma, '%e' ) ' 
']);
fprintf(fid0,['.bz ' p1to4 ' p5to8 ' 
]);
end
end
end
fclose(fid0);
if (MNA)
%Get the E field for one frequency (the first in samples)
clear Vin;
unix('rm dump.out');
unix('rm dump.m');
unix('rm peec.m');
cat Geometry.inp Part2MNA.inp > 4by4Array.inp;
~/peec/src/peec 4by4Array.inp > dump.out;
run('peec')
%peec.m contains the probe data, the frequencies (samples),
%the coordinates repr. min and max of the volume cells, Xmin and Xmax, and
%the normalized direction of each current, I.
fid0=fopen('dump.out', 'r'); %Open the dump file
fid1=fopen('dump.m', 'w'); %Make it an m file
while 1
inline=fgetl(fid0);
if length(inline)>=3 && strcmp(inline(1:3), 'x=[') || feof(fid0)
break
end
end
fprintf(fid1,[inline ' 
']);
close(fid0);close(fid1);
run('dump')
%dump.m contains the potentials and currents from the mna
x=x(size(x,2)-size(I,2)+1:size(x,2));
I=I.*x;
Vin=Vin(1,1)
Iin=1-Vin./50
f=samples(1);
[dBGain,S11]=EvalAntenna(Vin,Iin,Xmin,Xmax,I,f);
fitness=-abs(S11)+dBGain
return
else
unix('cat Geometry.inp Part2.inp > 4by4Array.inp');
~/peec/src/peec 4by4Array.inp';
peec
%peec.m contains the probe data, the frequencies (samples),
%the coordinates repr. min and max of the volume cells, Xmin and Xmax, and
%the normalized direction of each current, I.
Iin=1-Vin./50;
Zin=Vin./Iin;
figure(1)
subplot(211),plot(samples/1e9,real(Vin.*conj(Iin))),
xlabel('Freq. [GHz]'),ylabel('Power [W]'),grid on
subplot(212),plot(samples/1e9,abs(Iin)),xlabel('Freq. [GHz]'),
ylabel('Abs(Iin) [V]'),grid on
pause
a=find(abs(Iin)>0.5)&abs(gradient(abs(Iin)))<0.005); %Find the sections of resonance
c=[];
account=1;
ycount=1;
for count=1:length(a)-1 %Separate the sections into rows of c
43

```matlab
43
44 temp=a(count);
45 next=a(count+1);
46
47 if temp+1<next
48 c(ycount,xcount)=temp;
49 xcount=xcount+1;
50 ycount=1;
51 c(ycount,xcount)=next;
52 count=count+1;
53 else
54 c(ycount,xcount)=temp;
55 ycount=ycount+1;
56 end
57
58 if count==length(a)-1
59 c(ycount,xcount)=a(count+1);
60 end
61 end
62
63 a=[];
64 if length(c)>0
65 %Get the resonances in GHz
66 for resFreq=1:length(c(1,:))
67 a(resFreq)=find(max(abs(Iin(c(find(c(:,resFreq)>0),
68 ... resFreq])))=abs(Iin));
69 end
70 end
71 else
72 a=[]; %No resonance frequencies within the window of interest
73 end
74 fitness=0;
75 if length(a)==2
76 %Evaluate fitness only if there exist two resonance frequencies
77 fitness=1-abs(5/length(samples)*(a(1)-95))/5-
78 ... abs(5/length(samples)*(a(2)-202))/5;
79 end
80 if fitness<0
81 fitness=0;
82 end
83 end
84
85 MyFunction.m:
86 function fitness = MyFunction(b)
87 %Assuming length(b)=10
88 xbin =[]; ybin=[];
89 for count=1:length(b)/2
90 if b(count)==1
91 xbin=[xbin 1];
92 else
93 xbin=[xbin 0];
94 end
95 end
96 for count=length(b)/2+1:length(b)
97 if b(count)==1
98 ybin=[ybin 1];
99 else
100 ybin=[ybin 0];
101 end
102 end
103 [x,y]=mahagrid([-10:0:25,1:25])
104 spax=gray2dec(xbin);
105 spay=gray2dec(ybin);
106 w=sort(x(1,spay+1).*2*y(spax+1,1)).*eps;
107 fitness=abs(sin(r))/r;%Radial sinc function
108 if imag(fitness)~0;
109 fitness=0;
110 end
111
112 Field.m:
113 function E=Field(x,X1,X2,I,f)
114 %E = Field(x,X1,X2,I,f)
115 %Returns a vector of the E-field in cartesian coordinates
116 %x, point of observation in cartesian coordinates (3 by 1) [m]
117 %X1, min(x,y,z) for each volume cell (3 by n) [cm]
118 %X2, max(x,y,z) for each volume cell (3 by n) [cm]
119 %I, Current for each volume cell in cartesian coordinates. (3 by n) [A]
120 %f, The frequency of the excitation current (scalar) [Hz]
121 I=real(I); %Only a real current will produce a far-field.
122 for i=1:size(I,1)
123 E(i)=Field(x(i,1,1,1),I(i))
124 EXi=Field(x(i,1,1,1),I(i))
125 for j=1:size(1,2,1,1)
126 EX1i=Field(x(i,1,1,1),I(i))
127 end
128 end
129 end
```

%Note! The distances are relative to the volume cell and the observer

Epol=[0 0 0];

displacement=x(1)-Xprime(1,i);
ydisplacement=x(2)-Xprime(2,i);
zdisplacement=x(3)-Xprime(3,i);

RRel=sqrt(xdisplacement^2+ydisplacement^2+zdisplacement^2);

projRRel=sqrt(xdisplacement^2+ydisplacement^2);

sinThetaRel=projRRel/RRel;
cosThetaRel=zdisplacement/RRel;
sinPhiRel=ydisplacement/projRRel;
cosPhiRel=xdisplacement/projRRel;

%Er=Epol(1) always 0. Ephi field only a special case if
%observer is on the xy plane. For consistency only use currents

Epol(2)=j*Z0*(k*I(3,i)*Xlength(3,i)*exp(-j*k*RRel))/(4*pi*RRel)*sinThetaRel;

Epol(3)=-j*Z0*(k*I(1,i)*Xlength(1,i)*exp(-j*k*RRel))/(4*pi*RRel)*sinPhiRel;

Epol(3)=Epol(3)-j*Z0*(k*I(2,i)*Xlength(2,i)*exp(-j*k*RRel))/(4*pi*RRel)*cosPhiRel;

convSph2Cart=[sinThetaRel*cosPhiRel sinThetaRel*sinPhiRel cosThetaRel]';

E=E+convSph2Cart*Epol;

end

EvalAntenna.m:

function [dBGain,S11]=EvalAntenna(varargin)

% EvalAntenna(Vin,Iin,X1,X2,I,f)

Vin=varargin{1};
Iin=varargin{2};
X1=varargin{3};
X2=varargin{4};
I=varargin{5};
f=varargin{6};

R=3;

%Distance to the observer

Pin=real(Vin*conj(Iin))

if (Pin<0)
 error('Negative power');
 end

Z=Vin/Iin;

S11=(Z-75)/(Z+75);
%Assume a 75 ohm coaxial cable

%First, sweep the xy plane,
theta=[0.001:(pi/180):pi pi:-(pi/180):0.001];
phi=[ones(1,length(theta)/2)*pi*0.5 ones(1,length(theta)/2)*pi*1.5];

Ezy=[];

for count=1:length(theta)
%For every point of observation

Ezy=Field([R*sin(theta(count))*cos(phi(count)) R*sin(theta(count))*sin(phi(count)) R*cos(theta(count))] ,X1,X2,I,f);

end

Ezyabs=abs(Ezy).^2;

Enormzy=(sqrt(Ezyabs(1,:).^2+Ezyabs(2,:).^2+Ezyabs(3,:).^2));

Uzy=R^2.*1./(120*pi)*Enormzy;

%then, the zx plane

theta=[0.001:(pi/180):pi pi:-(pi/180):0.001];
phi=[ones(1,length(theta)/2)*pi*0 ones(1,length(theta)/2)*pi*1];

Ezx=[];

for count=1:length(theta)
%For every point of observation

Ezx=Field([R*sin(theta(count))*cos(phi(count)) R*sin(theta(count))*sin(phi(count)) R*cos(theta(count))] ,X1,X2,I,f);

end

Ezxabs=abs(Ezx).^2;

Enormzx=(sqrt(Ezxabs(1,:).^2+Ezxabs(2,:).^2+Ezxabs(3,:).^2));

Uzx=R^2.*1./(120*pi)*Enormzx;

Umax=max(max(Uzy),max(Uzx));

Gmax=4*pi*Umax/Pin

dBGain=10*log10(Gmax)

return
Bibliography


