

Chapter 2

Antenna Design Using Electromagnetic Simulations

In this chapter, we formulate the antenna design task as a nonlinear minimization problem. We introduce necessary notation, discuss typical objectives and constraints, and give a brief overview of conventional optimization techniques, including gradient-based and derivative-free methods, as well as metaheuristics. We also introduce the concept of the surrogate-based optimization (SBO) and discuss it on a generic level. More detailed exposition of SBO and SBO-related design techniques will be given in Chaps. 3 and 4.

2.1 Antenna Design Task as an Optimization Problem

Let $\mathbf{R}_f(\mathbf{x})$ denote a response of a high-fidelity (or fine) model of the antenna under design. For the rest of this book, we will assume that \mathbf{R}_f is obtained using accurate full-wave electromagnetic (EM) simulation. Typically, \mathbf{R}_f will represent evaluation of performance characteristics, e.g., reflection $|S_{11}|$ or gain over certain frequency band of interest. Vector $\mathbf{x} = [x_1 \ x_2 \ \dots \ x_n]^T$ represents designable parameters to be adjusted (e.g., geometry and/or material ones).

In some situations, individual components of the vector $\mathbf{R}_f(\mathbf{x})$ will be considered, and we will use the notation $\mathbf{R}_f(\mathbf{x}) = [R_f(\mathbf{x}, f_1) \ R_f(\mathbf{x}, f_2) \ \dots \ R_f(\mathbf{x}, f_m)]^T$, where $R_f(\mathbf{x}, f_k)$ is the evaluation of the high-fidelity model at a frequency f_k , whereas f_1 through f_m represent the entire discrete set of frequencies at which the model is evaluated.

The antenna design task can be formulated as the following nonlinear minimization problem (Koziel and Ogurtsov 2011a):

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} U(\mathbf{R}_f(\mathbf{x})) \tag{2.1}$$

where U is the scalar merit function encoding the design specifications, whereas \mathbf{x}^* is the optimum design to be found. The composition $U(\mathbf{R}_f(\mathbf{x}))$ will be referred to as the objective function. The function U is implemented so that a better design \mathbf{x} corresponds to a smaller value of $U(\mathbf{R}_f(\mathbf{x}))$. In antenna design, U is most often

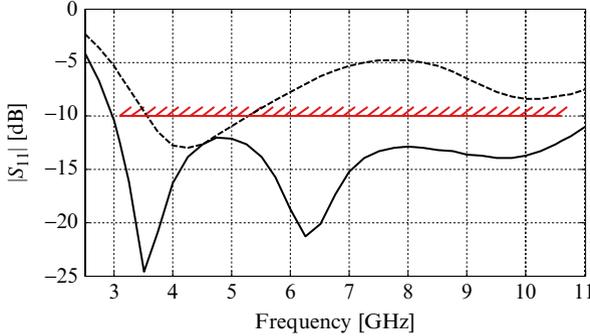


Fig. 2.1 Illustration of minimax design specifications, here, $|S_{11}| \leq -10$ dB for 3.1–10.6 GHz, marked with *thick horizontal line*. An example UWB antenna reflection response that does not satisfy our specifications (*dashed line*) (specification error, i.e., maximum violation of the requirements is about +5 dB) and another response that does satisfy the specifications (*solid line*)

implemented as a minimax function with upper (and/or lower) specifications. Figure 2.1 shows the example of minimax specifications corresponding to typical UWB requirements for the reflection response, i.e., $|S_{11}| \leq -10$ dB for 3.1–10.6 GHz. The value of $U(\mathbf{R}_f(\mathbf{x}))$ (also referred to as minimax specification error) corresponds to the maximum violation of the design specifications within the frequency band of interest.

To simplify notation, we will occasionally use the symbol $f(\mathbf{x})$ as an abbreviation for $U(\mathbf{R}_f(\mathbf{x}))$.

In reality, the problem (2.1) is always constrained. The following types of constraints can be considered:

- Lower and upper bounds for design variables, i.e., $l_k \leq x_k \leq u_k$, $k = 1, \dots, n$
- Inequality constraints, i.e., $c_{\text{ineq},l}(\mathbf{x}) \leq 0$, $l = 1, \dots, N_{\text{ineq}}$, where N_{ineq} is the number of constraints
- Equality constraints, i.e., $c_{\text{eq},l}(\mathbf{x}) = 0$, $l = 1, \dots, N_{\text{eq}}$, where N_{eq} is the number of constraints

Design constraints are usually introduced to make sure that the antenna structure that is to be evaluated by the EM solver is physically valid (e.g., certain parts of the structure do not overlap). Also, constraints can be introduced in order to ensure that the physical dimensions (length, width, area) of the antenna do not exceed certain assumed values.

In this book, geometry constraints such as those described above are handled explicitly. Other types of constraints, particularly those that emerge due to converting initially multi-objective design problem into single-objective one, are handled through penalty functions. It should be mentioned though that the literature offers efficient ways of explicit handling expensive constraints; see, e.g., Kazemi et al. (2011), Basudhar et al. (2012).

Figure 2.2 shows the simulation-driven design optimization flowchart. Typically, it is an iterative process where the designs found by the optimizer are verified by

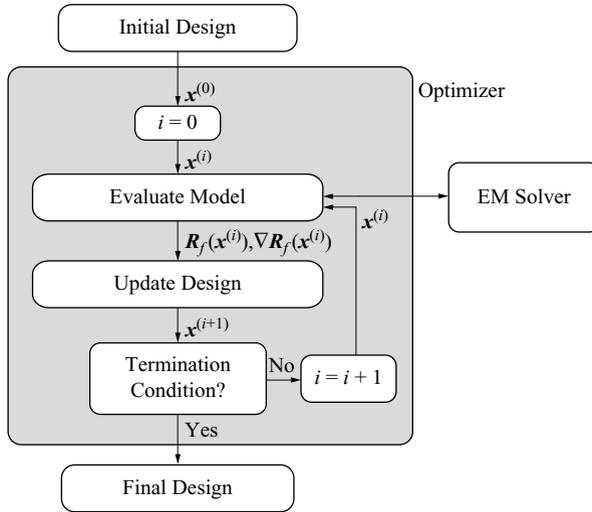


Fig. 2.2 Simulation-driven design through optimization. Generic optimization scheme is an iterative process where the new candidate designs are generated by the optimization algorithm and the high-fidelity model is evaluated through EM simulation for verification purposes and to provide the optimizer with information that allows searching for possible better designs. Depending on the type of the algorithm, the search process may be guided by the model response or (if available) by the response and its derivatives (gradient)

evaluating the high-fidelity model in the EM solver and—depending on a particular algorithm—the search process is guided either by the model response itself or the response of its gradients (if available). In Sects. 2.2–2.5, we briefly discuss conventional optimization approaches. In Chaps. 3 and 4, we discuss surrogate-based optimization methods, which are the main topic of this book.

2.2 Gradient-Based Optimization Methods

Gradient-based optimization techniques are the oldest and the most popular optimization methods (Nocedal and Wright 2000). In order to proceed toward the optimum design, they utilize derivative information of the objective function. Assuming that the objective $f(\mathbf{x})$ is sufficiently smooth (i.e., at least continuously differentiable), the gradient $\nabla f = [\partial f / \partial x_1 \ \partial f / \partial x_2 \ \dots \ \partial f / \partial x_n]^T$ gives the information about descent of f in the vicinity of the design at which the gradient is calculated. More specifically,

$$f(\mathbf{x} + \mathbf{h}) \cong f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot \mathbf{h} < f(\mathbf{x}) \tag{2.2}$$

for sufficiently small \mathbf{h} as long as $\nabla f(\mathbf{x}) \cdot \mathbf{h} < 0$. In particular $\mathbf{h} = -\nabla f(\mathbf{x})$ determines the direction of the steepest descent. In practice, using steepest descent as a search direction results in a poor performance of the optimization algorithm (Nocedal and

Wright 2000; Yang 2010). Better results are obtained using so-called conjugate-gradient method where the search direction is determined as a combination of the previous direction \mathbf{h}_{prev} and the current gradient, i.e.,

$$\mathbf{h} = -\nabla f(\mathbf{x}^i) + \gamma \mathbf{h}_{\text{prev}} \quad (2.3)$$

An example way of selecting the coefficient γ is a Fletcher-Reeves method with

$$\gamma = \frac{\nabla f(\mathbf{x})^T \nabla f(\mathbf{x})}{\nabla f(\mathbf{x}_{\text{prev}})^T \nabla f(\mathbf{x}_{\text{prev}})} \quad (2.4)$$

Having the search direction, the next design \mathbf{x}^{i+1} is determined from the current one \mathbf{x}^i as

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \alpha \cdot \mathbf{h} \quad (2.5)$$

Here, the choice of the step size $\alpha > 0$ is of great importance (Nocedal and Wright 2000), and finding it is referred to as a line search.

It is also possible to utilize second-order derivative information, which is characteristic to so-called Newton methods. Assuming f is at least twice continuously differentiable, one can consider a second-order Taylor expansion of f :

$$f(\mathbf{x} + \mathbf{h}) \cong f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot \mathbf{h} + \frac{1}{2} \mathbf{h} \cdot \mathbf{H}(\mathbf{x}) \cdot \mathbf{h} \quad (2.6)$$

where $\mathbf{H}(\mathbf{x})$ is the Hessian of f at \mathbf{x} , i.e., $\mathbf{H}(\mathbf{x}) = [\partial^2 f / \partial x_j \partial x_k]_{j,k=1,\dots,n}$. This means, given the current design \mathbf{x}^i being sufficiently close to the minimum of f , that the next approximation of the optimum can be determined as

$$\mathbf{x}^{i+1} = \mathbf{x}^i - [\mathbf{H}(\mathbf{x})]^{-1} \nabla f(\mathbf{x}) \quad (2.7)$$

If the starting point is sufficiently close to the optimum and the Hessian is positive definite (Yang 2010), the algorithm (2.7) converges very quickly to the (locally) optimal design. In practice, neither of these conditions is usually satisfied, so various types of damped Newton techniques are used, e.g., Levenberg-Marquardt method (Nocedal and Wright 2000). On the other hand, the Hessian of the objective function f is normally not available so quasi-Newton methods are used instead where the Hessian is approximated using various updating formulas (Nocedal and Wright 2000).

From the point of view of simulation-driven antenna design, the use of gradient-based methods is problematic mostly because of the high computational cost of accurate simulation and the fact that gradient-based methods normally require large number of objective function evaluations to converge, unless cheap way of obtaining sensitivity is utilized (e.g., through adjoints or automatic differentiation). Another problem is numerical noise that is always present in EM-based objective functions. Recently, adjoint sensitivity techniques have become available in some

commercial EM solvers (CST 2013; HFSS 2010), which may revive the interest in this type of methods for antenna design because they allow calculation of sensitivity at little or no extra cost compared to a regular EM simulation of the antenna structure. On the other hand, automatic differentiation is usually not an option because source codes are not accessible whenever commercial solvers are utilized. It should also be mentioned that gradient-based methods exploiting a trust-region framework are usually more efficient than those based, e.g., on line search so that using trust region (Conn et al. 2000) is recommended whenever possible.

2.3 Derivative-Free Optimization Methods

In many situations, gradient-based search may not be a good option. This is particularly the case when derivative information is not available or expensive to compute (e.g., through finite differentiation of an expensive objective function). Also, if the objective function is noisy (which is typical for responses obtained from EM simulation) then the gradient-based search does not perform well.

Optimization techniques that do not use derivative data in the search process are referred to as derivative-free methods. Formally speaking metaheuristics (Sect. 2.4) as well as many surrogate-based approaches (Chaps. 3 and 4) also fall into this category. In this section, however, we only mention the basic idea of the local search methods. Figure 2.3 illustrates the concept of the pattern search (Kolda et al. 2003), where the search of the objective function minimum is restricted to the rectangular grid and explores a grid-restricted vicinity of the current design. Failure in making the step improve the current design leads to refining the grid size and allowing

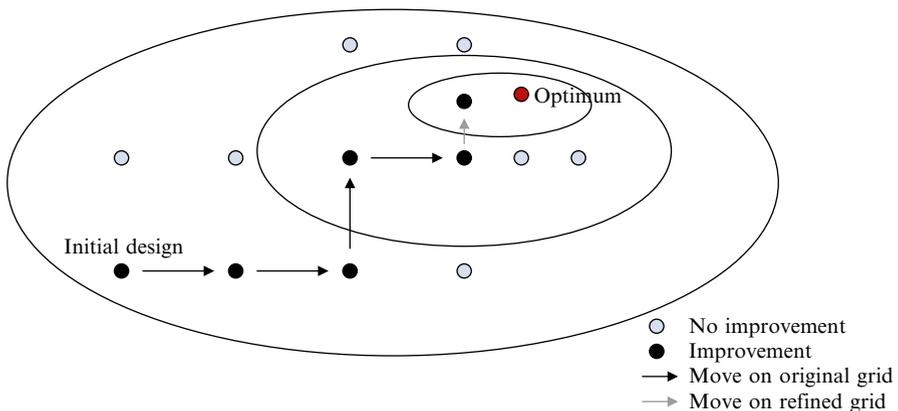


Fig. 2.3 The concept of pattern search. The search is based on exploratory movements restricted to the rectangular grid around the initial design. Upon failure of making the successful move, the grid is refined to allow smaller steps. The actual implementations of pattern search routines also use more sophisticated strategies (e.g., grid-restricted line search)

smaller steps. Various variants of the pattern search methods are available (see, e.g., Torczon 1997; Kolda et al. 2003). With sufficiently large size of the initial grid, these techniques can be used to perform a quasi-global search.

One of the most famous derivative-free methods is the Nelder-Mead algorithm (Yang 2010) also referred to as the simplex method. Its search process is based on moving the vertices of the simplex in the design space in such a way that the vertex corresponding to the worst (i.e., highest) value of the objective function is replaced by the new one at the location where the objective function value is expected to be improved.

Pattern search and similar methods are usually robust although their convergence is relatively slow compared to gradient-based routines. Their fundamental advantage is in the fact that they do not use derivative information and, even more importantly, they are quite immune to the numerical noise. An excellent and mathematically rigorous treatment of derivative-free optimization techniques, including model-based trust-region derivative-free methods, can be found in Conn et al. (2009). Many pattern search methods and their extensions possess mathematically rigorous convergence guarantees (Conn et al. 2009). An interesting extension of pattern search to constrained black-box optimization is Mesh Adaptive Direct Search (MADS) (Audet and Dennis 2006).

2.4 Metaheuristics and Global Optimization

Metaheuristics are global search methods that are based on observation of natural processes (e.g., biological or social systems). Most metaheuristics process sets (or populations) of potential solutions to the optimization problem at hand in a way that these solutions (also called individuals) interact with each other so that the optimization process is capable to avoid getting stuck in local optima and converge—with reasonable probability—to a globally optimal solution of the problem. At the same time, metaheuristics can handle noisy, non-differentiable, and discontinuous objective functions.

The most popular types of metaheuristic algorithms include genetic algorithms (GAs) (Goldberg 1989), evolutionary algorithms (EAs) (Back et al. 2000), evolution strategies (ES) (Back et al. 2000), particle swarm optimizers (PSO) (Kennedy et al. 2001), differential evolution (DE) (Storn and Price 1997), and, more recently, firefly algorithm (Yang 2010). A famous example of metaheuristic algorithm that processes a single solution rather than a population of individuals is simulated annealing (Kirkpatrick et al. 1983).

The typical flow of the metaheuristic algorithm is the following (here, P is the set of potential solutions to the problem at hand, also referred to as a population):

1. Initialize population P .
2. Evaluate population P .
3. Select parent individuals S from P .

4. Apply recombination operators to create a new population P from parent individuals S .
5. Apply mutation operators to introduce local perturbations in individuals of P .
6. If termination condition is not satisfied, go to 2.
7. End.

Initialization of the population is usually random. In the next stage, each individual is evaluated, and its corresponding value of the objective function determines its “fitness.” An important step is selection of the subset of individuals to form a new population. Depending on the algorithm, the selection can be deterministic (pick up the best ones only, ES) or partially random (probability of being selected depends on the fitness value but there is a chance even for poor individuals, EAs). In some algorithms, such as PSO or DE, there is no selection at all (i.e., individuals are modified from iteration to iteration but never die). There are two types of operations that are used to modify individuals: exploratory ones (e.g., crossover in EAs or ES) and exploitative ones (e.g., mutation in GAs). Exploratory operators combine information contained in the parent individuals to create a new one. For example, in case of an evolutionary algorithm with natural (floating point) representation, a new individual c can be created as a convex combination of the parents p_1 and p_2 , i.e., $c = \alpha p_1 + (1 - \alpha)p_2$, where $0 < \alpha < 1$ is a random number. These types of operators allow making large “steps” in the design space and, therefore, explore new and promising regions. Exploitative operators introduce small perturbations (e.g., $p \leftarrow p + r$, where r is a random vector selected according to a normal probability distribution with zero mean and certain, problem-dependent variance). These operators allow exploitation of a given region of the design space improving local search properties of the algorithm. In some of the more recent algorithms, e.g., PSO, the difference between both types of operators is not that clear (modification of the individual may be based on the best solution found so far by that given individual as well as the best solution found by the entire population).

A common feature of metaheuristics is that they normally require a large number of objective function evaluations to converge. Typical population size is anything between 10 and 100, whereas the number of iteration may be a few dozen to a few hundred. Also, their performance may be heavily dependent on the values of control parameters, which may not be easy to determine beforehand. Finally, because they are stochastic methods, a solution obtained in each run of the algorithm will be generally different from the previous one. From the point of view of antenna design, metaheuristics are attractive approaches for problems where evaluation time of the objective function is not of a primary concern and when multiple local optima are expected. For that reason, metaheuristics are mostly used for solving antenna array optimization problems with non-coupled radiators, in particular, for pattern synthesis (e.g., Ares-Pena et al. 1999; Bevelacqua and Balanis 2007; Li et al. 2008). It should also be mentioned that the problem of high computational cost can be partially alleviated by surrogate-assisted metaheuristics (e.g., Ong et al. 2003; Emmerich et al. 2006; Zhou et al. 2007; Jin 2011; Parno et al. 2012; Loshchilov et al. 2012; Regis 2013a, b), where metaheuristic optimization is combined with response surface modeling of the expensive objective function.

2.5 Challenges of Conventional Optimization Toward Design Using Surrogate Models

The optimization methods considered in this chapter attempt to solve the design problem (2.1) directly. In this sense, we refer to these techniques as conventional ones. As explained in Sect. 2.1 and Fig. 2.2, the direct approach requires that each new candidate design produced by the optimizer is verified by performing EM simulation of the underlying antenna structure. Because each high-fidelity simulation is already computationally expensive, conventional optimization with its large number of objective function evaluations may be prohibitive. Numerical noise that is inherent to EM simulations poses additional problems, particularly for gradient-based methods. Consequently, application of conventional off-the-shelf optimization algorithms for EM-based antenna design often results in failures. As a result, although almost all commercial simulation tools offer certain built-in optimization capabilities (e.g., CST 2013; FEKO 2012), many designers rely on repetitive parameter sweeps and own expert knowledge that allow them to find at least satisfactory designs in reasonable time.

The surrogate-based optimization concept that is a main topic of this book attempts to address this problem by replacing direct optimization of the high-fidelity model, with optimization of its cheap and analytically tractable representation referred to as a surrogate model. As indicated in the following chapters, it is possible to construct and exploit such representations in such a way that—with occasional reference to the high-fidelity model—a satisfactory design can be found at a fraction of a computational effort required by conventional optimization.



<http://www.springer.com/978-3-319-04366-1>

Antenna Design by Simulation-Driven Optimization

Koziel, S.; Ogurtsov, S.

2014, IX, 141 p. 94 illus., 45 illus. in color., Softcover

ISBN: 978-3-319-04366-1