1 Introduction

Optimization is a human instinct. People constantly seek to improve their lives and the systems that surround them. Optimization is intrinsic in biology, as exemplified by the evolution of species. Birds optimize their wings' shape in real time, and dogs have been shown to find optimal trajectories. Even more broadly, many laws of physics relate to optimization, such as the principle of minimum energy. As Leonhard Euler once wrote, "nothing at all takes place in the universe in which some rule of maximum or minimum does not appear."

The term *optimization* is often used to mean "improvement", but mathematically, it is a much more precise concept: finding the *best* possible solution by changing variables that can be controlled, often subject to constraints. Optimization has a broad appeal because it is applicable in all domains and because of the human desire to make things better. Any problem where a decision needs to be made can be cast as an optimization problem.

Although some simple optimization problems can be solved analytically, most practical problems of interest are too complex to be solved this way. The advent of numerical computing, together with the development of optimization algorithms, has enabled us to solve problems of increasing complexity.

By the end of this chapter you should be able to:

1. Understand the design optimization process.
2. Formulate an optimization problem.
3. Identify key characteristics to classify optimization problems and optimization algorithms.
4. Select an appropriate algorithm for a given optimization problem.

Optimization problems occur in various areas, such as economics, political science, management, manufacturing, biology, physics, and engineering. This book focuses on the application of numerical opti-

mization to the design of engineering systems. Numerical optimization first emerged in *operations research*, which deals with problems such as deciding on the price of a product, setting up a distribution network, scheduling, or suggesting routes. Other optimization areas include optimal control and machine learning. Although we do not cover these other areas specifically in this book, many of the methods we cover are useful in those areas.

Design optimization problems abound in the various engineering disciplines, such as wing design in aerospace engineering, process control in chemical engineering, structural design in civil engineering, circuit design in electrical engineering, and mechanism design in mechanical engineering. Most engineering systems rarely work in isolation and are linked to other systems. This gave rise to the field of *multidisciplinary design optimization* (MDO), which applies numerical optimization techniques to the design of engineering systems that involve multiple disciplines.

In the remainder of this chapter, we start by explaining the design optimization process and contrasting it with the conventional design process (Section 1.1). Then we explain how to formulate optimization problems and the different types of problems that can arise (Section 1.2). Because design optimization problems involve functions of different types, these are also briefly discussed (Section 1.3). (A more detailed discussion of the numerical models used to compute these functions is deferred to Chapter 3.) We then provide an overview of the different optimization algorithms, highlighting the algorithms covered in this book and linking to the relevant sections (Section 1.4). We connect algorithm types and problem types by providing guidelines for selecting the right algorithm for a given problem (Section 1.5). Finally, we introduce the notation used throughout the book (Section 1.6).

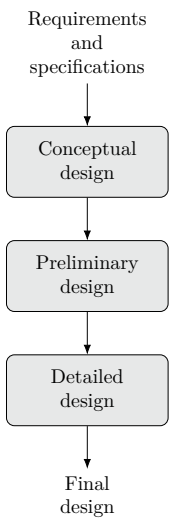

Fig. 1.1 Design phases.

1.1 Design Optimization Process

Engineering design is an iterative process that engineers follow to develop a product that accomplishes a given task. For any product beyond a certain complexity, this process involves teams of engineers and multiple stages with many iterative loops that may be nested. The engineering teams are formed to tackle different aspects of the product at different stages.

The design process can be divided into the sequence of phases shown in Fig. 1.1. Before the design process begins, we must determine the requirements and specifications. This might involve market research, an analysis of current similar designs, and interviews with potential

customers. In the conceptual design phase, various concepts for the system are generated and considered. Because this phase should be short, it usually relies on simplified models and human intuition. For more complicated systems, the various subsystems are identified. In the preliminary design phase, a chosen concept and subsystems are refined by using better models to guide changes in the design, and the performance expectations are set. The detailed design phase seeks to complete the design down to every detail so that it can finally be manufactured. All of these phases require iteration within themselves. When severe issues are identified, it may be necessary to "go back to the drawing board" and regress to an earlier phase. This is just a high-level view; in practical design, each phase may require multiple iterative processes.

Design optimization is a tool that can replace an iterative design process to accelerate the design cycle and obtain better results. To understand the role of design optimization, consider a simplified version of the conventional engineering design process with only one iterative loop, as shown in Fig. 1.2 (top). In this process, engineers make decisions at every stage based on intuition and background knowledge.

Each of the steps in the conventional design process includes human decisions that are either challenging or impossible to program into computer code. Determining the product specifications requires engineers to define the problem and do background research. The design cycle must start with an initial design, which can be based on past designs or a new idea. In the conventional design process, this initial design is analyzed in some way to evaluate its performance. This could involve numerical modeling or actual building and testing. Engineers then evaluate the design and decide whether it is good enough or not based on the results.* If the answer is no—which is likely to be the case for at least the first few iterations—the engineer changes the design based on intuition, experience, or trade studies. When the design is finalized when it is deemed satisfactory.

*The evaluation of a given design in engineering is often called the *analysis*. Engineers and computer scientists also refer to it as *simulation*.

The design optimization process can be represented using a flow diagram similar to that for the conventional design process, as shown in Fig. 1.2 (bottom). The determination of the specifications and the initial design are no different from the conventional design process. However, design optimization requires a formal formulation of the optimization problem that includes the design variables that are to be changed, the objective to be minimized, and the constraints that need to be satisfied. The evaluation of the design is strictly based on numerical values for the objective and constraints. When a rigorous optimization algorithm is used, the decision to finalize the design is made only when the current

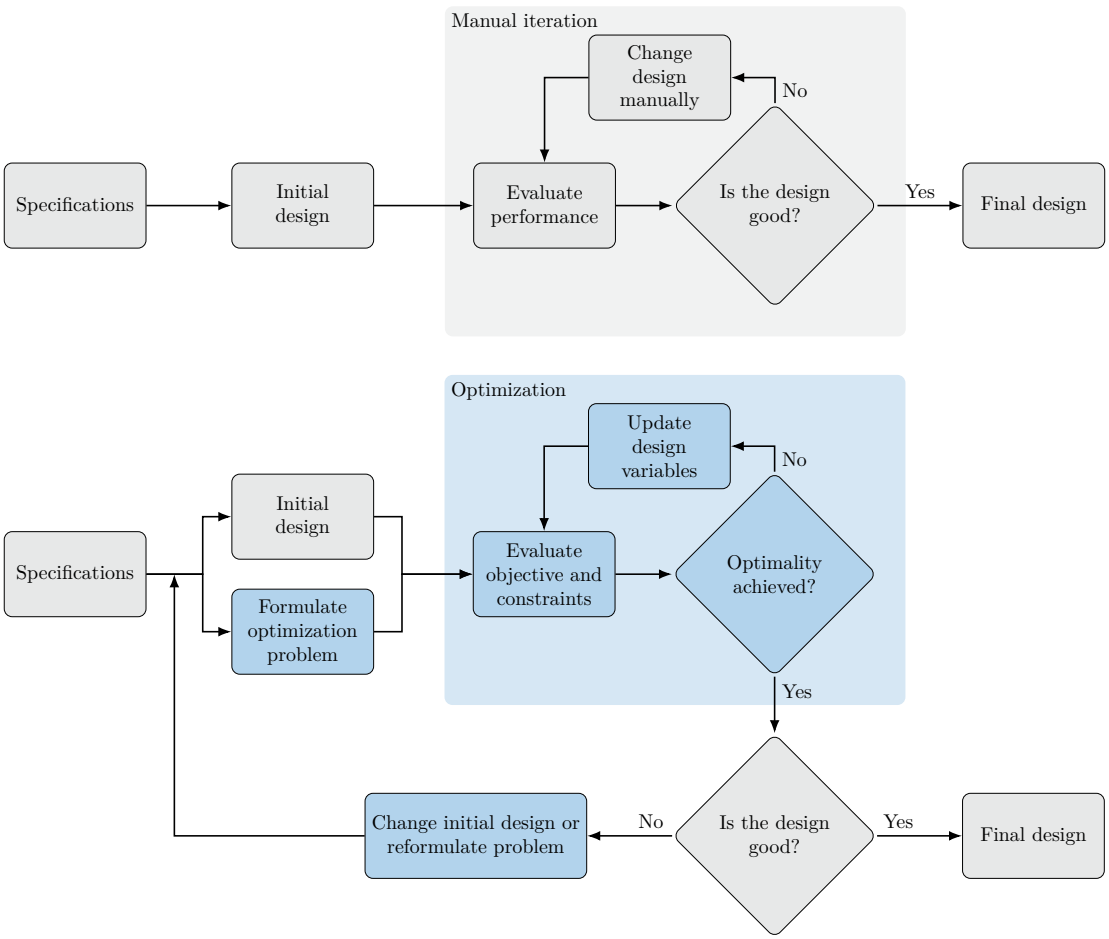

Fig. 1.2 Conventional (top) versus design optimization process (bottom).

design satisfies the optimality conditions that ensure that no other design "close by" is better. The design changes are made automatically by the optimization algorithm and do not require intervention from the designer.

This automated process does not usually provide a "push-button" solution; it requires human intervention and expertise (often more expertise than in the traditional process). Human decisions are still needed in the design optimization process. Before running an optimization, in addition to determining the specifications and initial design, engineers need to formulate the design problem. This requires expertise in both the subject area and numerical optimization. The designer must decide what the objective is, which parameters can be changed, and which constraints must be enforced. These decisions have profound effects on the outcome, so it is crucial that the designer formulates the optimization problem well.

After running the optimization, engineers must assess the design because it is unlikely that the first formulation yields a valid and practical design. After evaluating the optimal design, engineers might decide to reformulate the optimization problem by changing the objective function, adding or removing constraints, or changing the set of design variables. Engineers might also decide to increase the models' fidelity if they fail to consider critical physical phenomena, or they might decide to decrease the fidelity if the models are too expensive to evaluate in an optimization iteration.

Post-optimality studies are often performed to interpret the optimal design and the design trends. This might be done by performing parameter studies, where design variables or other parameters are varied to quantify their effect on the objective and constraints. Validation of the result can be done by evaluating the design with higher-fidelity simulation tools, by performing experiments, or both. It is also possible to compute post-optimality sensitivities to evaluate which design variables are the most influential or which constraints drive the design. These sensitivities can inform where engineers might best allocate resources to alleviate the driving constraints in future designs.

Design optimization can be used in any of the design phases shown in Fig. 1.1, where each phase could involve running one or more design optimizations. We illustrate several advantages of design optimization in Fig. 1.3, which shows the notional variations of system performance, cost, and uncertainty as a function of time in design. When using optimization, the system performance increases more rapidly compared with the conventional process, achieving a better end result in a shorter total time. As a result, the cost of the design process is lower. Finally, the uncertainty in the performance reduces more rapidly as well.

Considering multiple disciplines or components using MDO amplifies the advantages illustrated in Fig. 1.3. The central idea of MDO is to consider the interactions between components using coupled models while simultaneously optimizing the design variables from the various components. In contrast, sequential optimization optimizes one component at a time. Even when interactions are considered, sequential optimization might converge to a suboptimal result (see Section 13.1 for more details and examples).

In this book, we tend to frame problems and discussions in the context of engineering design. However, the optimization methods are general and are used in other applications that may not be design problems, such as optimal control, machine learning, and regression. In other words, we mean "design" in a general sense, where variables are changed to optimize an objective.

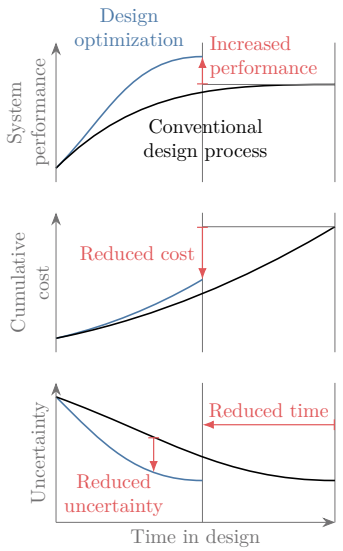

Fig. 1.3 Compared with the conventional design process, MDO increases the system performance, decreases the design time, reduces the total cost, and reduces the uncertainty at a given point in time.

1.2 Optimization Problem Formulation

The design optimization process requires the designer to translate their intent to a mathematical statement that can then be solved by an optimization algorithm. Developing this statement has the added benefit that it helps the designer better understand the problem. Being methodical in the formulation of the optimization problem is vital because *the optimizer tends to exploit any weaknesses you might have in your formulation or model*. An inadequate problem formulation can either cause the optimization to fail or cause it to converge to a mathematical optimum that is undesirable or unrealistic from an engineering point of view—the proverbial "right answer to the wrong question".

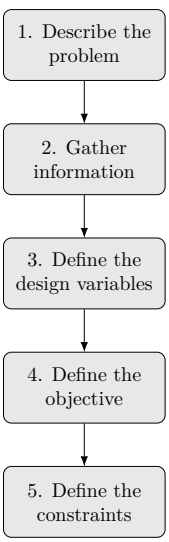

Fig. 1.4 Steps in optimization problem formulation.

To formulate design optimization problems, we follow the procedure outlined in Fig. 1.4. The first step requires writing a description of the design problem, including a description of the system, and a statement of all the goals and requirements. At this point, the description does not necessarily involve optimization concepts and is often vague.

The next step is to gather as much data and information as possible about the problem. Some information is already specified in the problem statement, but more research is usually required to find all the relevant data on the performance requirements and expectations. Raw data might need to be processed and organized to gather the information required for the design problem. The more familiar practitioners are with the problem, the better prepared they will be to develop a sound formulation to identify eventual issues in the solutions.

At this stage, it is also essential to identify the analysis procedure and gather information on that as well. The analysis might consist of a simple model or a set of elaborate tools. All the possible inputs and outputs of the analysis should be identified, and its limitations should be understood. The computational time for the analysis needs to be considered because optimization requires repeated analysis.

It is usually impossible to learn everything about the problem before proceeding to the next steps, where we define the design variables, objective, and constraints. Therefore, information gathering and refinement are ongoing processes in problem formulation.

1.2.1 Design Variables

The next step is to identify the variables that describe the system, the *design variables*,* which we represent by the column vector:

$$x = [x_1, x_2, \ldots, x_{n_x}] \, . \tag{1.1}$$

*Some texts call these *decision variables* or simply *variables*.

This vector defines a given design, so different vectors x correspond to different designs. The number of variables, n_x, determines the problem's dimensionality.

The design variables must not depend on each other or any other parameter, and the optimizer must be free to choose the elements of x independently. This means that in the analysis of a given design, the variables must be input parameters that remain fixed throughout the analysis process. Otherwise, the optimizer does not have absolute control of the design variables. Another possible pitfall is to define a design variable that happens to be a linear combination of other variables, which results in an ill-defined optimization problem with an infinite number of combinations of design variable values that correspond to the same design.

The choice of variables is usually not unique. For example, a square shape can be parametrized by the length of its side or by its area, and different unit systems can be used. The choice of units affects the problem's scaling but not the functional form of the problem.

The choice of design variables can affect the functional form of the objective and constraints. For example, some nonlinear relationships can be converted to linear ones through a change of variables. It is also possible to introduce or eliminate discontinuities through the choice of design variables.

A given set of design variable values defines the system's design, but whether this system satisfies all the requirements is a separate question that will be addressed with the constraints in a later step. However, it is possible and advisable to define the space of allowable values for the design variables based on the design problem's specifications and physical limitations.

The first consideration in the definition of the allowable design variable values is whether the design variables are *continuous* or *discrete*. Continuous design variables are real numbers that are allowed to vary continuously within a specified range with no gaps, which we write as

$$\underline{x}_i \leq x_i \leq \overline{x}_i, \quad i = 1, \ldots, n_x \,, \tag{1.2}$$

where $\underline{x}$ and $\overline{x}$ are lower and upper bounds on the design variables, respectively. These are also known as *bound constraints* or *side constraints*. Some design variables may be unbounded or bounded on only one side.

When all the design variables are continuous, the optimization problem is said to be continuous.[†] Most of this book focuses on algorithms that assume continuous design variables.

[†]This is not to be confused with the continuity of the objective and constraint functions, which we discuss in Section 1.3.

When one or more variables are allowed to have discrete values, whether real or integer, we have a discrete optimization problem. An example of a discrete design variable is structural sizing, where only components of specific thicknesses or cross-sectional areas are available. Integer design variables are a special case of discrete variables where the values are integers, such as the number of wheels on a vehicle. Optimization algorithms that handle discrete variables are discussed in Chapter 8.

We distinguish the design variable bounds from constraints because the optimizer has direct control over their values, and they benefit from a different numerical treatment when solving an optimization problem. When defining these bounds, we must take care not to unnecessarily constrain the design space, which would prevent the optimizer from achieving a better design that is realizable. A smaller allowable range in the design variable values should make the optimization easier. However, design variable bounds should be based on actual physical constraints instead of being artificially limited. An example of a physical constraint is a lower bound on structural thickness in a weight minimization problem, where otherwise, the optimizer will discover that negative sizes yield negative weight. Whenever a design variable converges to the bound at the optimum, the designer should reconsider the reasoning for that bound and make sure it is valid. This is because designers sometimes set bounds that limit the optimization from obtaining a better objective.

At the formulation stage, we should strive to list as many independent design variables as possible. However, it is advisable to start with a small set of variables when solving a problem for the first time and then gradually expand the set of design variables.

Some optimization algorithms require the user to provide initial design variable values. This initial point is usually based on the best guess the user can produce. This might be an already good design that the optimization refines further by making small changes. Another possibility is that the initial guess is a bad design or a "blank slate" that the optimization changes significantly.

Example 1.1 Design variables for wing design

Consider a wing design problem where the wing planform shape is rectangular. The planform could be parametrized by the span (b) and the chord (c), as shown in Fig. 1.5, so that $x = [b, c]$. However, this choice is not unique. Two other variables are often used in aircraft design: wing area (S) and wing aspect ratio (AR), as shown in Fig. 1.6. Because these variables are not independent ($S = bc$ and $AR = b^2/S$), we cannot just add them to the set

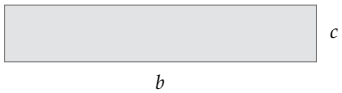

Fig. 1.5 Wingspan (b) and chord (c).

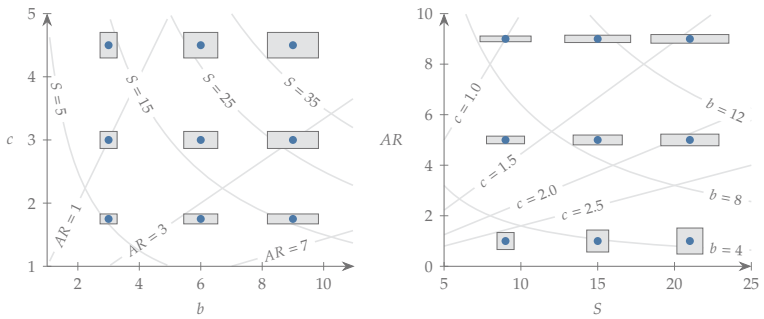

Fig. 1.6 Wing design space for two different sets of design variables, $x = [b, c]$ and $x = [S, AR]$.

of design variables. Instead, we must pick any two variables out of the four to parametrize the design because we have four possible variables and two dependency relationships.

For this wing, the variables must be positive to be physically meaningful, so we must remember to explicitly bound these variables to be greater than zero in an optimization. The variables should be bound from below by small positive values because numerical models are probably not prepared to take zero values. No upper bound is needed unless the optimization algorithm requires it.

Tip 1.1 Use splines to parameterize curves

Many problems that involve shapes, functional distributions, and paths are sometimes implemented with a large number of discrete points. However, these can be represented more compactly with splines. This is a commonly used technique in optimization because reducing the number of design variables often speeds up an optimization with little if any loss in the model parameterization fidelity. Figure 1.7 shows an example spline describing the shape of a turbine blade. In this example, only four design variables are used to represent the curved shape.

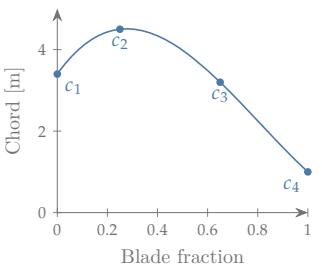

Fig. 1.7 Parameterizing the chord distribution of a wing or turbine blade using a spline reduces the number of design variables while still allowing for a wide range of shape changes.

1.2.2 Objective Function

To find the best design, we need an *objective function*, which is a quantity that determines if one design is better than another. This function must be a scalar that is computable for a given design variable vector x. The objective function can be minimized or maximized, depending on the problem. For example, a designer might want to minimize the weight or cost of a given structure. An example of a function to be maximized could be the range of a vehicle.

The convention adopted in this book is that the objective function, f, is to be *minimized*. This convention does not prevent us from maximizing a function because we can reformulate it as a minimization problem by finding the minimum of the negative of f and then changing the sign, as follows:

$$\max[f(x)] = -\min[-f(x)]\,. \tag{1.3}$$

This transformation is illustrated in Fig. 1.8.‡

‡Inverting the function $(1/f)$ is another way to turn a maximization problem into a minimization problem, but it is generally less desirable because it alters the scale of the problem and could introduce a divide-by-zero problem.

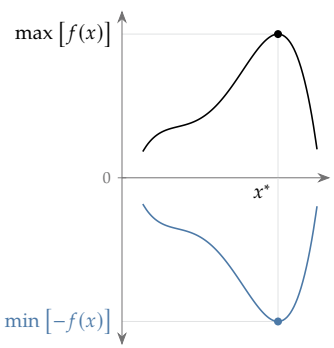

Fig. 1.8 A maximization problem can be transformed into an equivalent minimization problem.

The objective function is computed through a numerical model whose complexity can range from a simple explicit equation to a system of coupled implicit models (more on this in Chapter 3).

The choice of objective function is crucial for successful design optimization. If the function does not represent the true intent of the designer, it does not matter how precisely the function and its optimum point are computed—the mathematical optimum will be non-optimal from the engineering point of view. A bad choice for the objective function is a common mistake in design optimization.

The choice of objective function is not always obvious. For example, minimizing the weight of a vehicle might sound like a good idea, but this might result in a vehicle that is too expensive to manufacture. In this case, manufacturing cost would probably be a better objective. However, there is a trade-off between manufacturing cost and the performance of the vehicle. It might not be obvious which of these objectives is the most appropriate one because this trade-off depends on customer preferences. This issue motivates *multiobjective optimization*, which is the subject of Chapter 9. Multiobjective optimization does not yield a single design but rather a range of designs that settle for different trade-offs between the objectives.

Experimenting with different objectives should be part of the design exploration process (this is represented by the outer loop in the design optimization process in Fig. 1.2). Results from optimizing the "wrong" objective can still yield insights into the design trade-offs and trends for the system at hand.

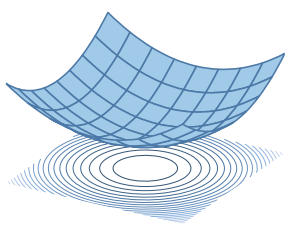

Fig. 1.9 A function of two variables ($f = x_1^2 + x_2^2$ in this case) can be visualized by plotting a three-dimensional surface or contour plot.

In Ex. 1.1, we have the luxury of being able to visualize the design space because we have only two variables. For more than three variables, it becomes impossible to visualize the design space. We can also visualize the objective function for two variables, as shown in Fig. 1.9. In this figure, we plot the function values using the vertical axis, which results in a three-dimensional surface. Although plotting the surface might provide intuition about the function, it is not possible to locate the points accurately when drawing on a two-dimensional surface.

Another possibility is to plot the contours of the function, which are lines of constant value, as shown in Fig. 1.10. We prefer this type