UNIVERSIDADE BEIRA INTERIOR

Introduction

Otimização em Engenharia (15235)

2º Ciclo/Mestrado em Engenharia Aeronáutica

2024

Pedro V. Gamboa

Departamento de Ciências Aeroespaciais

Faculdade de Engenharia

0. Topics

- Terminology, problem statement and classification of optimization problems
- Optimization in engineering, multidisciplinary design, and aerospace applications
- Methods and algorithms

1. Concepts in optimization problems

- Optimization is fundamental in human activities.
- Even nature tries to optimize its processes: by using less energy for flight, by taking the shortest root to a destination, by minimizing total energy for stability.
- “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 […]. Any problem where a decision needs to be made can be cast as an optimization problem.” (Martins et Ning, 2021)
- “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.” (Martins et Ning, 2021)

1. Concepts in optimization problems

- Optimization problems occur in many areas:
  - Economics
  - Political Science
  - Management
  - Manufacturing
  - Biology
  - Physics
  - Engineering
- The focus in this course will be on the optimization of engineering systems, even though the methods described may be used and are useful in other areas.

1. Concepts in optimization problems

- Design optimization problems are common in the various engineering disciplines:
  - wing design in aerospace engineering
  - process control in chemical engineering
  - structural design in civil engineering
  - circuit design in electrical engineering
  - 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.

1.1. Design optimization process

- Engineering design is an iterative process that engineers use to develop a product developed for a given task.
- For complex products, the design process involves teams of engineers with different expertise and multiple phase which may be iterative and be nested.
- The design process is divided into a sequence of phases.
- Fig. 1.01 shows a high-level schematic of the design process.

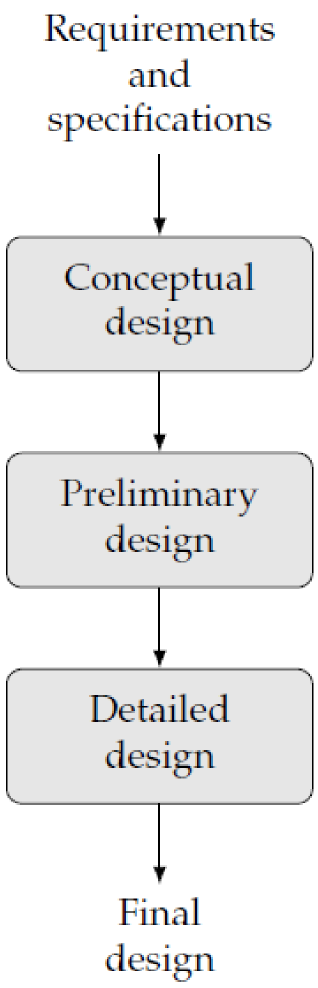


Figure 1.01 Design process.

Source: Martins et Ning, 2021

1.1. Design optimization process

- Design optimization is a tool that can be used to replace an iterative design process to accelerate the design cycle and obtain better results.
- Fig. 1.02 shows a classical simplified single iterative loop design process. In this process, engineers make decisions at every stage based on intuition and background knowledge.

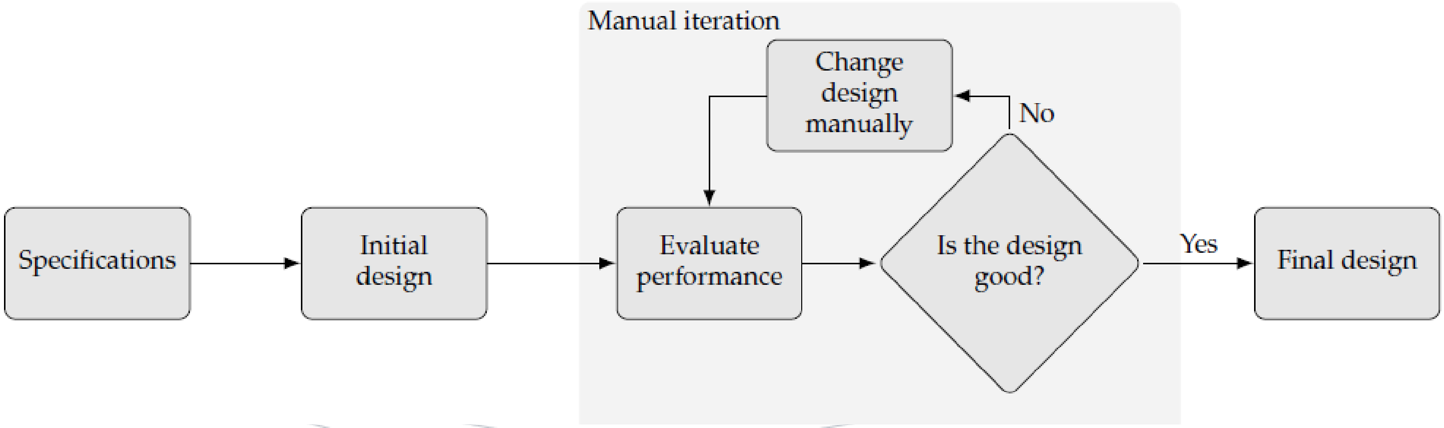


Figure 1.02 Classical engineering design process.

Source: Martins et Ning, 2021

1.1. Design optimization process

- At each iteration, the design must be evaluated.
- 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 as shown in Fig. 1.02.
- 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.

1.1. Design optimization process

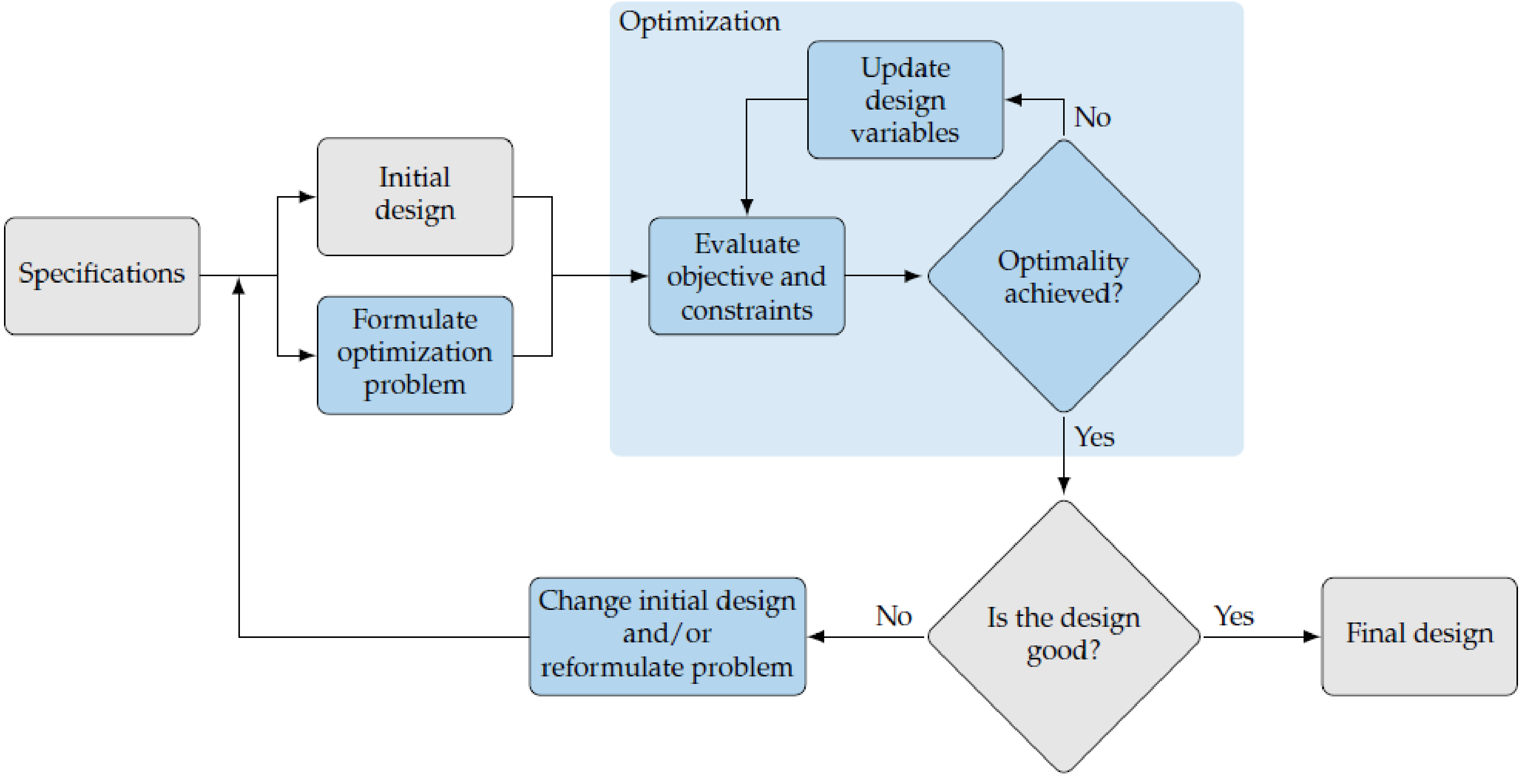


Figure 1.03 Design optimization process.

Source: Martins et Ning, 2021

1.1. Design optimization process

- When a rigorous optimization algorithm is used, the decision to finalize the design is made only when the current design satisfies the optimality conditions that ensure that no other design in the vicinity 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.

1.1. Design optimization process

- 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 the optimization is complete, the designer must assess the results, because there is no guarantee the obtained final the design is the required one.
- It may be necessary to reformulate the optimization problem or to change (increase or reduce) the fidelity of the analysis tools.

1.1. Design optimization process

- 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 the optimization is complete, the designer must assess the results, because there is no guarantee the obtained final the design is the required one.
- It may be necessary to reformulate the optimization problem or to change (increase or reduce) the fidelity of the analysis tools.

1.1. Design optimization process

- Post-optimality studies may also be used to interpret the optimal design and assess 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.
- 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.

1.1. Design optimization process

- Several advantages of design optimization arise, as shown in Fig. 1.04, which shows the notional variations of system performance, cost, and uncertainty as a function of time in design.
- The gains are increased in MDO problems.

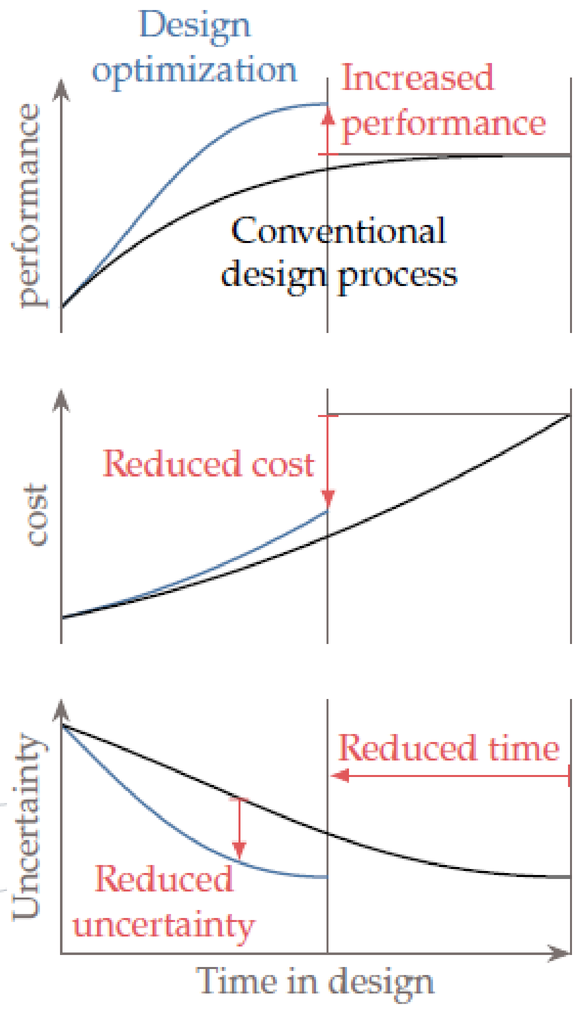


Figure 1.04 Advantages of design optimization over classical design process.

Source: Martins et Ning, 2021

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”.

1.2. Optimization problem formulation

- Fig. 1.05 outlines the sequence to formulate the optimization problem.
- 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.

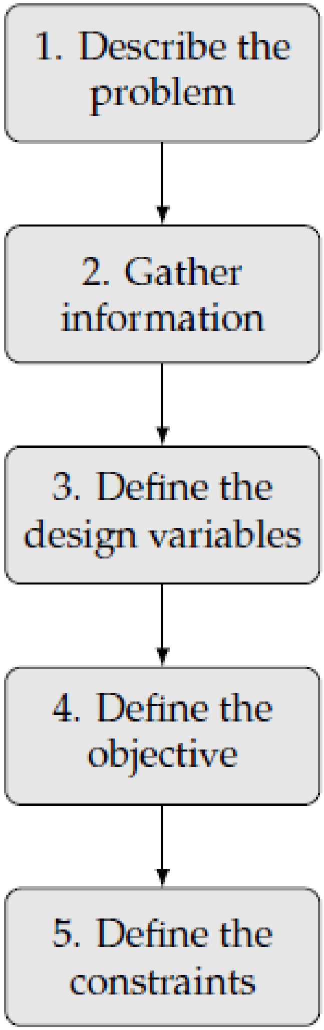


Figure 1.05 Steps in optimization problem formulation.

Source: Martins et Ning, 2021

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

$$x = \left[x_1, x_1, \ldots, x_{n_x}\right]^T \qquad (1.01)$$

- 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 components 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.

1.2.1. Design variables

- 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

$$x_{l,i} \leq x_i \leq x_{u,i}; \quad i = 1, \ldots, n_x \qquad (1.02)$$

- These are 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.

1.2.1. Design variables

- 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.
- A small 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.

1.2.1. Design variables

- 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.

1.2.1. Design variables

Example 1.1: Design variables for a wing design.

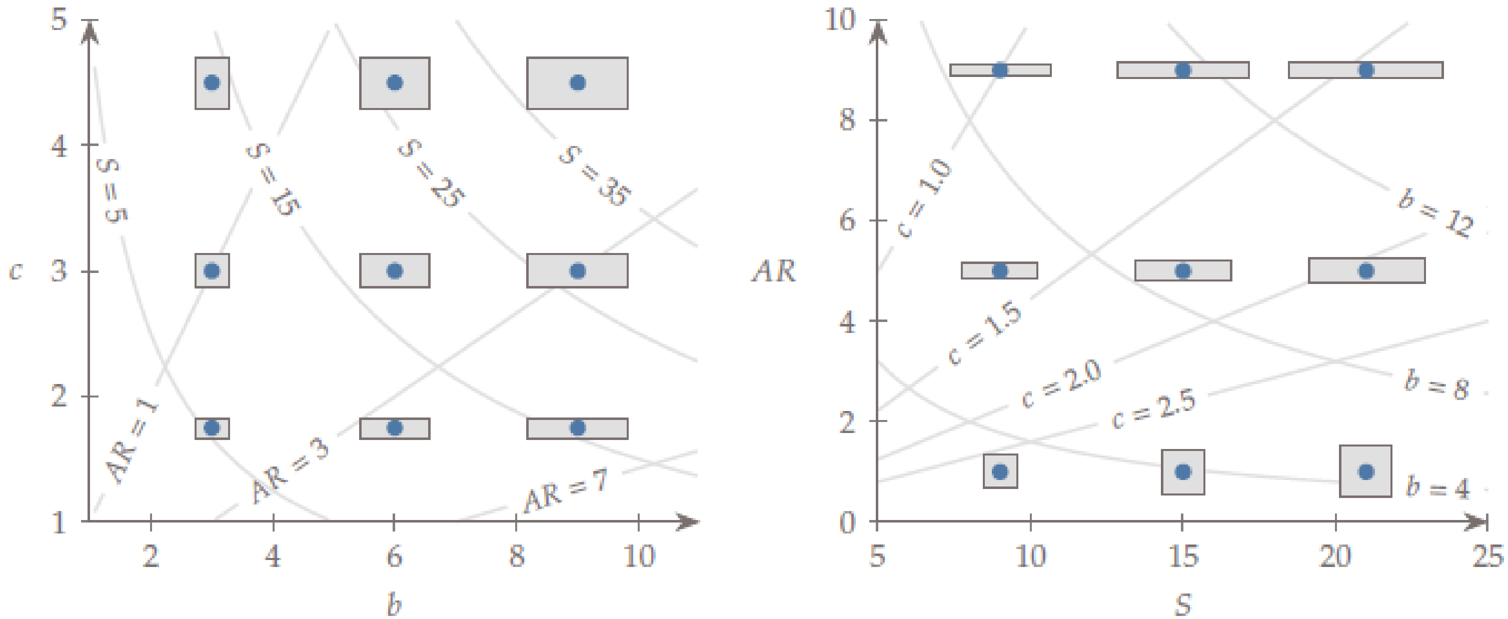


Figure 1.06 Design variables in two options: $x=[b,c]^T$ or $x=[S,AR]^T$.

Source: Martins et Ning, 2021

1.2.2. Objective function

- To find the best design, a quantifiable criterion is necessary to determine if one design is better than another – the objective function.
- The objective 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.
- Mos often, the objective function f is to be minimized.

1.2.2. Objective function

- This 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

$$\max[f(x)] = -\min[-f(x)] \qquad (1.03)$$

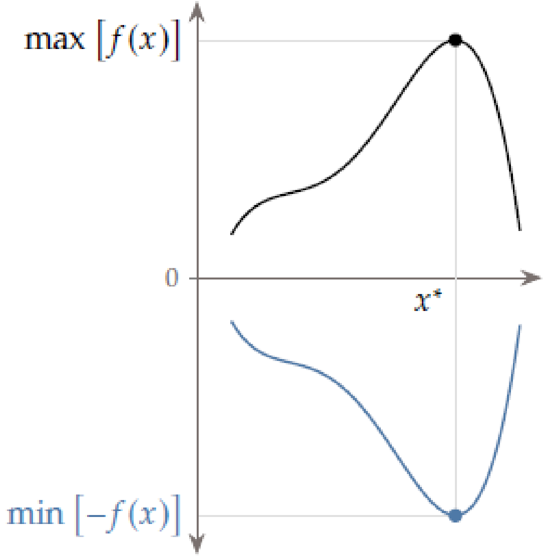


Figure 1.07 Maximize *f* or minimize -*f*.

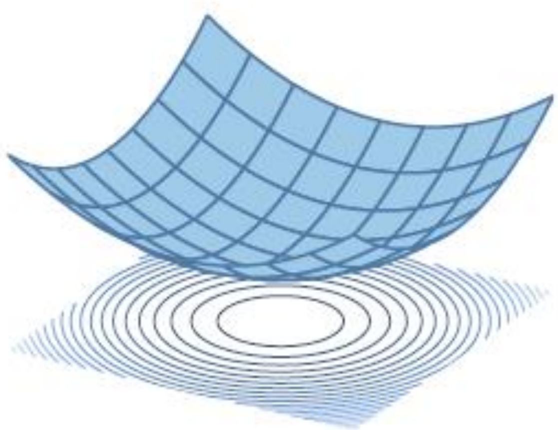
Source: Martins et Ning, 2021

1.2.2. Objective function

- The computation of f is performed through a numerical model.
- The choice of the objective function is crucial to optimize the problem at hand.
- If the objective function does not represent well the problem or it does not capture the nuances of the problem, the designer will not obtain a good result, even if the function is very precise.
- Such example could be by using weight minimization of manufacturing costs minimization of a given aircraft design.
- Sometimes, multiobjective optimization problems are convenient.
- Trying different optimization objectives is part of the design exploration process. This helps select the best objective.

1.2.2. Objective function

- Contour plots can be used to visualize the objective function in 2D design spaces.
- Larger dimensionality of the design spaces are not easy to represent.



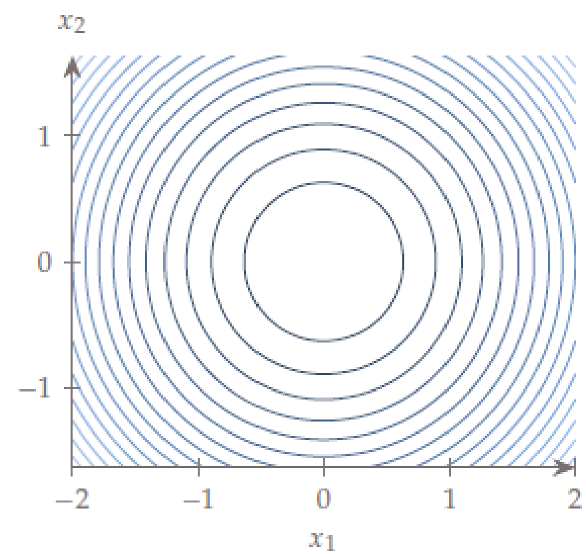


Figure 1.08 Function $f = x_1^2 + x_2^2$, in surface representation and contour representation.

Source: Martins et Ning, 2021

1.2.2. Objective function

Example 1.2: Objective function for a wing design.

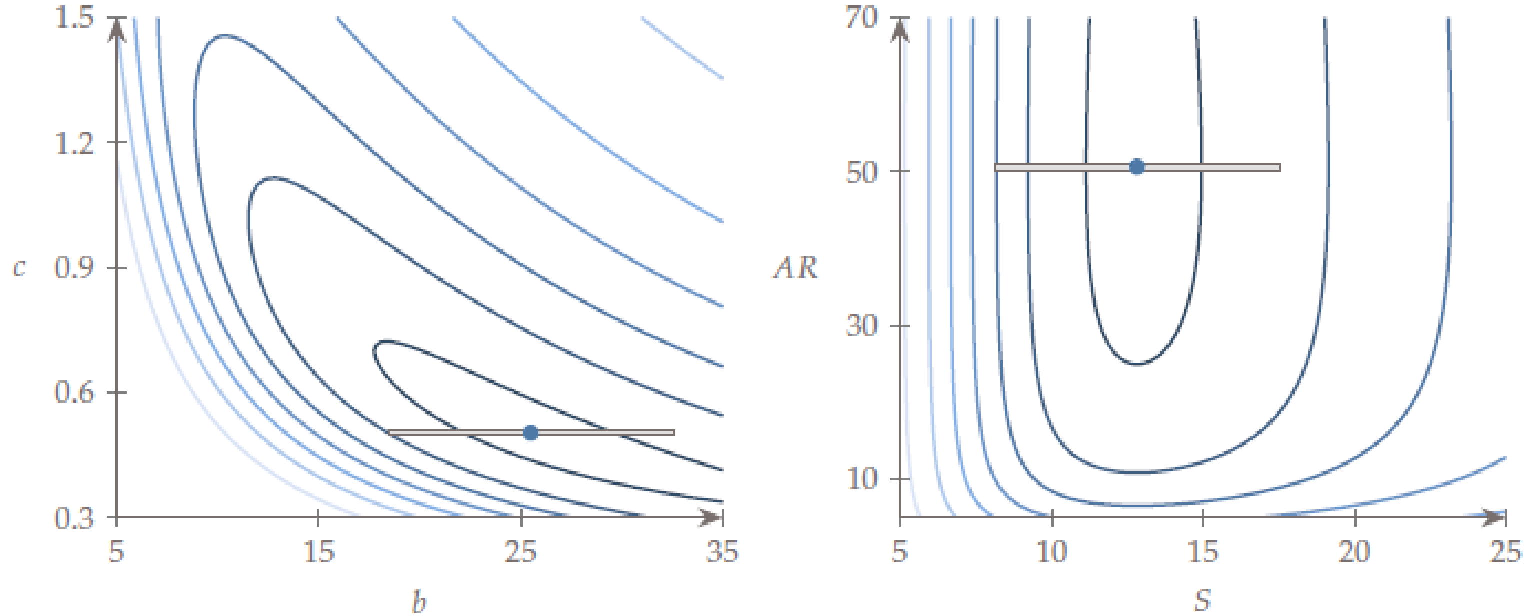


Figure 1.09 Power minimization for two design variables' sets. The optimal wing is the same for both cases, but the functional form of the objective is simplified in the one on the right.

Source: Martins et Ning, 2021

1.2.3. Constraints

- The vast majority of practical design optimization problems require the enforcement of constraints.
- These are functions of the design variables that we want to restrict in some way. Like the objective function, constraints are computed through a model whose complexity can vary widely.
- The feasible region is the set of points that satisfy all constraints.
- We seek to minimize the objective function within this feasible design space.
- When we restrict a function to being equal to a fixed value, we call this an equality constraint, denoted by $h(x)=0$.
- When the function is required to be less than or equal to a certain value, we have an inequality constraint, denoted by $g(x)\leq 0$

1.2.3. Constraints

- Although we use "less or equal" by convention, you should be aware that some other texts and software programs use "greater or equal" instead.
- There is no loss of generality with either convention because we can always multiply the constraint by -1 to convert between the two.
- Inequality constraints can be active or inactive at the optimum point.
- An active inequality constraint means that $g(x^*)=0$, whereas for an inactive one, $g(x^*)<0$.
- If a constraint is inactive at the optimum, this constraint could have been removed from the problem with no change in its solution, as illustrated in Fig. 1.10.

1.2.3. Constraints

- In this case, constraints g_2 and g_3 can be removed without affecting the solution of the problem.
- Furthermore, active constraints (g_1 in this case) can equivalently be replaced by equality constraints.

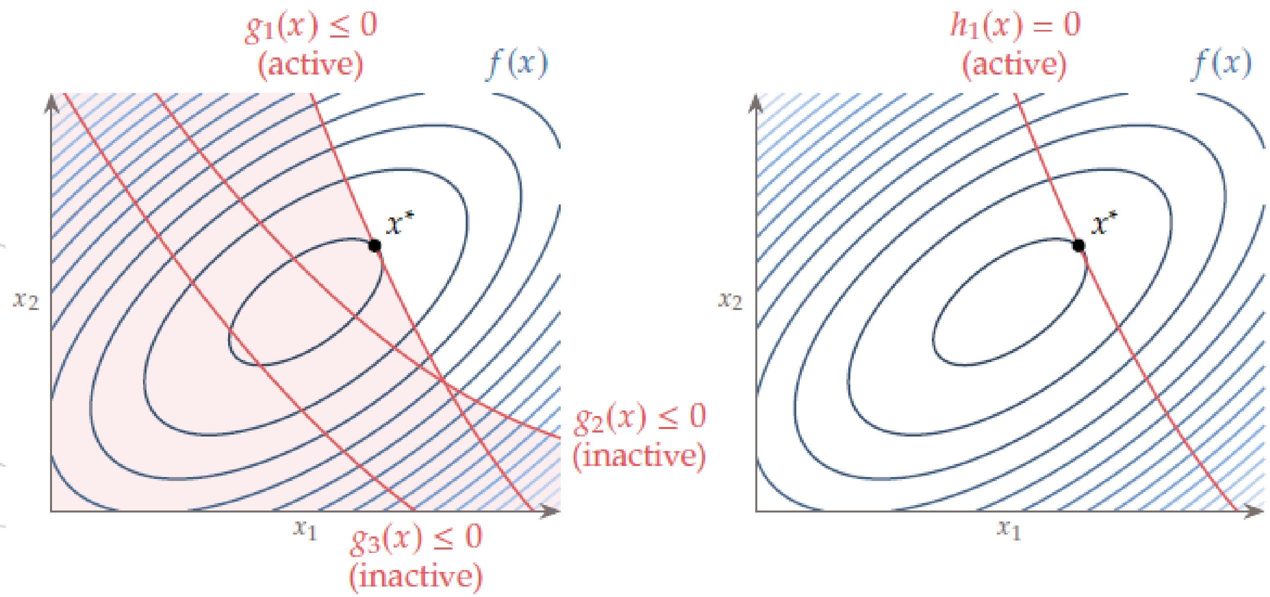


Figure 1.10 Two-dimensional problem with two inactive and one active constraint.

Source: Martins et Ning, 2021

1.2.3. Constraints

- It is possible to overconstrain a problem, rendering a non-feasible solution.
- This can result from programming errors or can happen at the problem formulation stage.
- When it is not possible to satisfy all constrains it may be necessary to omit or relax some of them.
- To prevent this, the number of independent equality constraints must be less than or equal to the design variables ($n_h \leq n_x$).
- There is no limit on the inequality constraints, but the sum of active inequality constraints with the equality constrains must still be less than or equal to the number of design variables.
- The feasible region grows as the number of constraints reduces, and the objective function usually improves.

1.2.3. Constraints

Example 1.3: Constraints for a wing design.

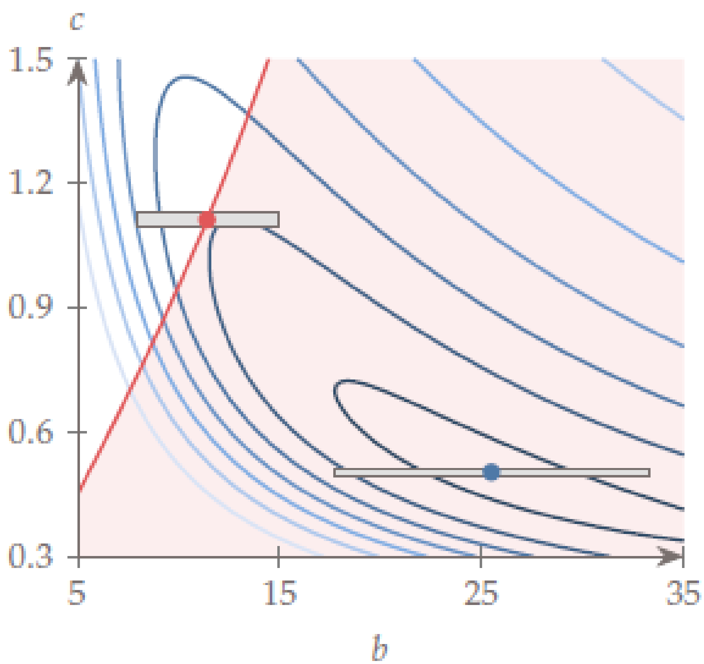


Figure 1.11 Minimum-power wing with a constraint on wing bending stress compared with the unconstrained solution.

Source: Martins et Ning, 2021

1.2.4. Optimization problem statement

- Now that we have discussed the definition of design variables, the objective function, and constraints, we can put them all together in an optimization problem statement.
- In words, this statement is as follows: *minimize the objective function by varying the design variables within their bounds subject to the constraints*.
- Mathematically, we write this as follows

$$
\begin{array}{lll}
\text{minimize} & f(x) & \\
\text{by varying} & x_{l,i} \leq x_i \leq x_{u,i} & i = 1, \ldots, n_x \\
\text{subjected to} & g_j(x) \leq 0 & j = 1, \ldots, n_g \\
 & h_l(x) = 0 & l = 1, \ldots, n_h
\end{array}
\qquad (1.04)
$$

1.2.4. Optimization problem statement

- All single-objective, continuous optimization problems can be written in this form.
- Although our target applications are engineering design problems, many other problems can be stated in this form, and thus, the methods covered here can be used to solve those problems.
- The values of the objective and constraint functions for a given set of design variables are computed through the analysis, which consists of one or more numerical models.
- The analysis must be fully automatic so that multiple optimization cycles can be completed without human intervention, as shown in Fig. 1.12.
- The optimizer usually requires an initial design x_0 and then queries the analysis for a sequence of designs until it finds the optimum design, x^*.

1.2.4. Optimization problem statements

- When the optimizer queries the analysis for a given *x*, for most
- methods, the constraints do not have to be feasible.
- The optimizer is responsible for changing *x* so that the constraints are satisfied.
- The objective and constraint functions must depend on the design variables; if a function does not depend on any variable in the whole domain, it can be ignored and should not appear in the problem statement.

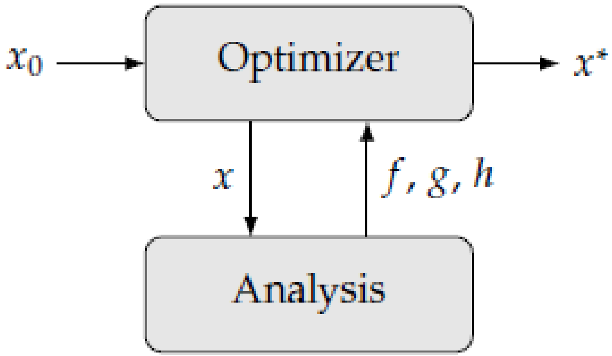


Figure 1.12 The analysis computes the objective (*f*) and constraint values (*g*,*h*) for a given set of design variables (*x*).

Source: Martins et Ning, 2021

1.2.4. Optimization problem statements

- Ideally, f , g, and h should be computable for all values of x that
- make physical sense.
- Lower and upper design variable bounds should be set to avoid nonphysical designs as much as possible
- Even after taking this precaution, models in the analysis sometimes fail to provide a solution. A good optimizer can handle such eventualities easily.
- Determining an appropriate set of design variables, objective, and constraints is a crucial aspect of the outer loop shown in Fig. 1.03, which requires human expertise in engineering design and numerical optimization.

1.3. Optimization problem classification

- To choose the most appropriate optimization algorithm for solving a given optimization problem, we must classify the optimization problem and know how its attributes affect the efficacy and suitability of the available optimization algorithms.
- This is important because no optimization algorithm is efficient or even appropriate for all types of problems.
- We classify optimization problems based on two main aspects:
  - the problem formulation
  - the characteristics of the objective and constraint functions

1.3. Optimization problem classification

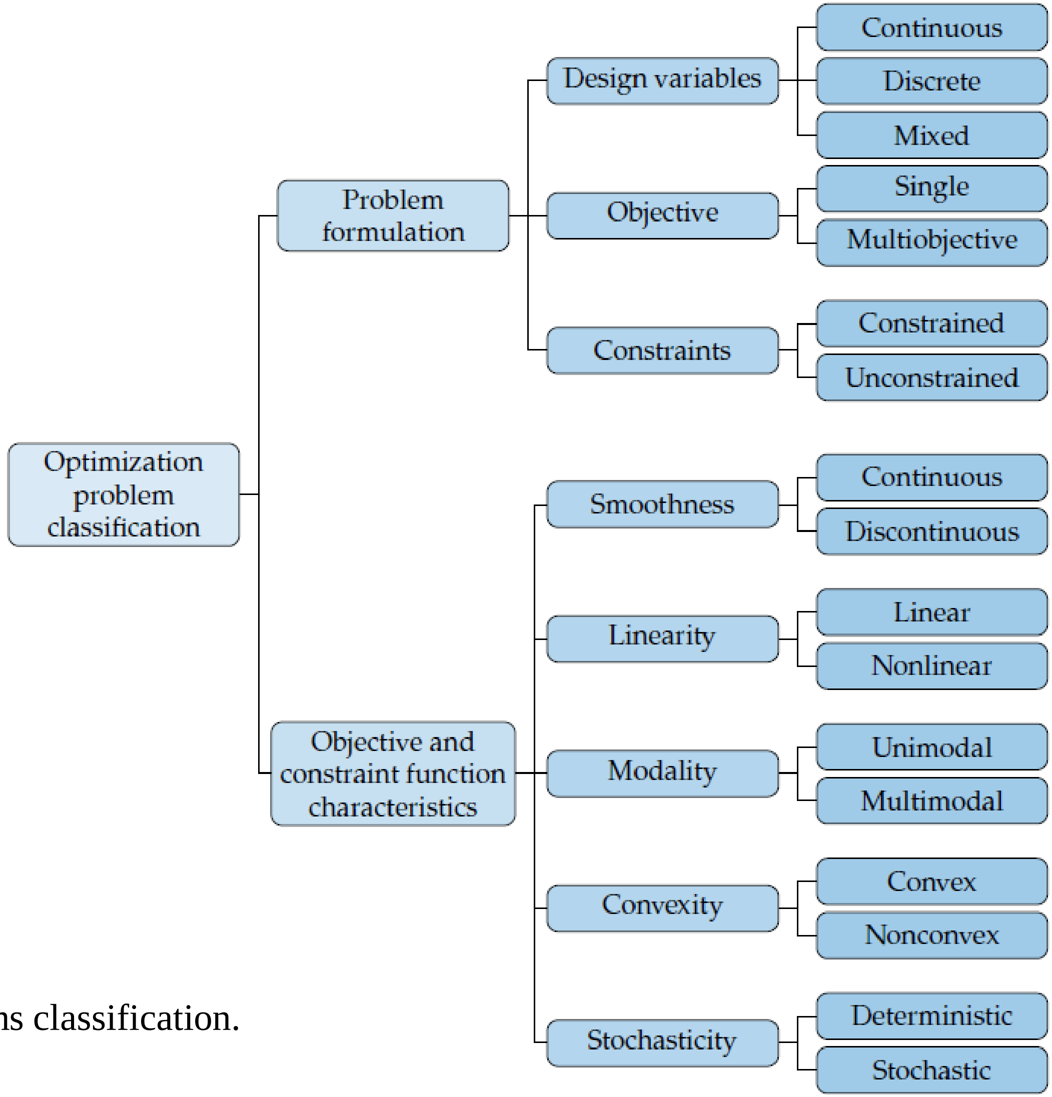


Figure 1.13 Optimization problems classification.

Source: Martins et Ning, 2021

1.3. Optimization problem classification

- Here the function is viewed as a "black box" – a computation for which we only see inputs (including the design variables) and outputs (including objective and constraints), as illustrated in Fig. 1.14.
- When dealing with black-box models, there is limited, or no understanding of the modelling and numerical solution process used to obtain the function values.
- We can still characterize the functions based purely on their outputs.
- The black-box view is common in real-world applications. This might be because the source code is not provided, the modelling methods are not described, or simply because the user does not bother to understand them.

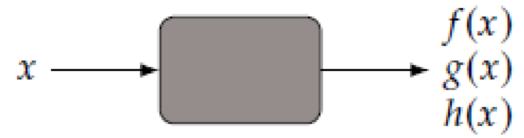


Figure 1.14 An analysis model is considered a black box when only the inputs and outputs are known.

Source: Martins et Ning, 2021

1.3.1. Smoothness

- The degree of function smoothness with respect to variations in the design variables depends on the continuity of the function values and their derivatives.
- When the value of the function varies continuously, the function is said to be C^0 continuous.
- If the first derivatives also vary continuously, then the function is C^1 continuous, and so on.
- A function is smooth when the derivatives of all orders vary continuously everywhere in its domain.

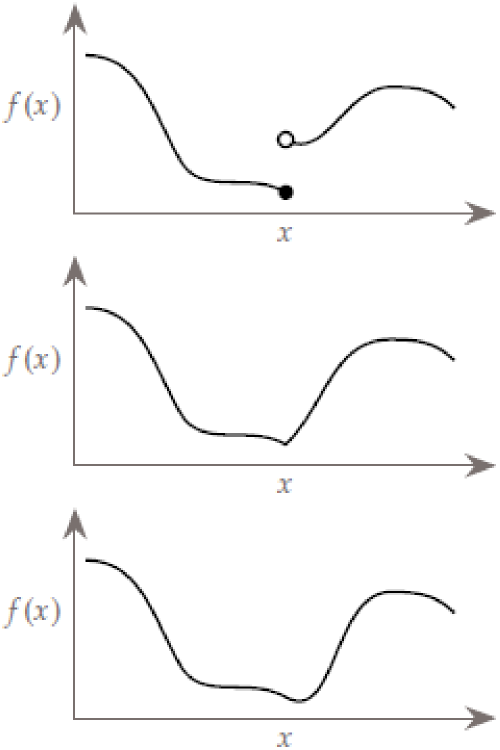


Figure 1.15 Discontinuous function (top), C^0 continuous function (middle), and C^1 continuous function (bottom).

Source: Martins et Ning, 2021

1.3.2. Linearity

- The functions of interest could be linear or nonlinear.
- When both the objective and constraint functions are linear, the optimization problem is known as a linear optimization problem.
- These problems are easier to solve than general nonlinear ones, and those are dealt with in Chapter 2.
- The first numerical optimization algorithms were developed to solve linear optimization problems, and there are many applications in operations. An example of a linear optimization problem is shown in Fig. 1.16.

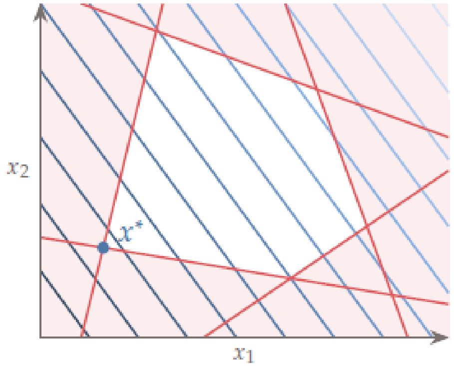


Figure 1.16 Example of linear optimization problem with two variables.

Source: Martins et Ning, 2021

1.3.2. Linearity

- An optimization problem consisting of a quadratic objective function and linear constraints is a quadratic optimization problem.
- Although many problems can be formulated as linear or quadratic problems, most engineering design problems are nonlinear.

1.3.3. Multimodality and Convexity

- Functions can be either unimodal or multimodal.
- Unimodal functions have a single minimum, whereas multimodal functions have multiple minima.
- When we find a minimum without knowledge of whether the function is unimodal or not, we can only say that it is a local minimum; that is, this point is better than any point within a small neighbourhood.
- When we know that a local minimum is the best in the whole domain (because we somehow know that the function is unimodal), then this is also the global minimum, as illustrated in Fig. 1.17.
- Sometimes, the function might be flat around the minimum, in which case we have a weak minimum.

1.3.3. Multimodality and Convexity

- It is difficult to prove a function is unimodal.
- However, it is much easier to prove multimodality – all is necessary is to find two distinct local minima.
- Often, we need not be too concerned about the possibility of multiple local minima.
- From an engineering design point of view, achieving a local optimum that is better than the initial design is already a useful result.

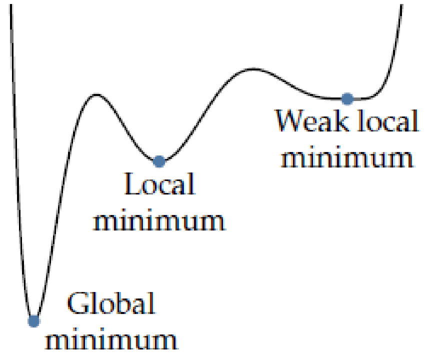


Figure 1.17 Types of minima.

Source: Martins et Ning, 2021

1.3.3. Multimodality and Convexity

- Convexity is a concept related to multimodality.
- A function is convex if all line segments connecting any two points in the function lie above the function and never intersect it.
- Convex functions are always unimodal.
- Also, all multimodal functions are nonconvex, but not all unimodal functions are convex (see Fig. 1.18).
- Convex optimization seeks to minimize convex functions over convex sets.

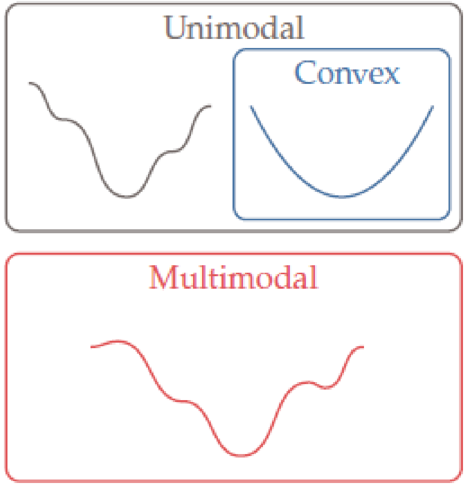


Figure 1.18 Multimodal functions have multiple minima, whereas unimodal functions have only one minimum. All multimodal functions are nonconvex, but not all unimodal functions are convex.

Source: Martins et Ning, 2021

1.3.3. Multimodality and Convexity

- Like linear optimization, convex optimization is another subfield of numerical optimization with many applications
- When the objective and constraints are convex functions, we can use specialized formulations and algorithms that are much more efficient than general nonlinear algorithms to find the global optimum.

1.3.4. Deterministic versus Stochastic

- Some functions are inherently stochastic.
- A stochastic model will yield different function values for repeated evaluations with the same input (Fig. 1.19).
- For example, the numerical value from a roll of dice is a stochastic function.
- Stochasticity can also arise from deterministic models when the inputs are subject to uncertainty.

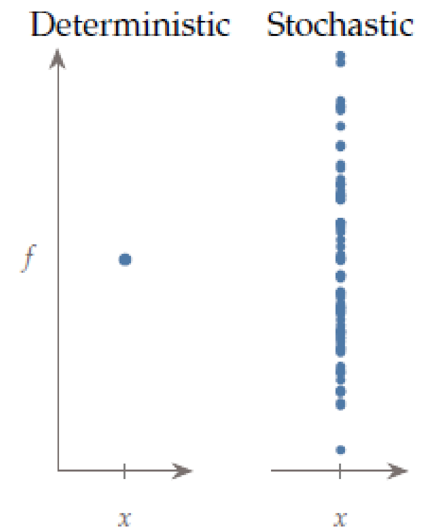


Figure 1.19 Deterministic functions yield the same output when evaluated repeatedly for the same input, whereas stochastic functions do not.

Source: Martins et Ning, 2021

1.3.4. Deterministic versus Stochastic

- The input variables are then described as probability distributions, and their uncertainties need to be propagated through the model.
- For example, the bending stress in a beam may follow a deterministic model, but the beam's geometric properties may be subject to uncertainty because of manufacturing deviations.

1.4. Optimization algorithms

- No single optimization algorithm is effective or even appropriate for all possible optimization problems.
- This is why it is important to understand the problem before deciding which optimization algorithm to use.
- By "effective" algorithm, we mean that the algorithm is capable of solving the problem, and secondly, it does so reliably and efficiently.
- Fig. 1.20 lists the attributes for the classification of optimization algorithms, which we cover in more detail in the following discussion.
- These attributes are often amalgamated, but they are independent, and any combination is possible.

1.4. Optimization algorithms

- When multiple models are involved, we also need to consider how the models are coupled, solved, and integrated with the optimizer.
- These considerations lead to different MDO architectures, which may involve multiple levels of optimization problems.

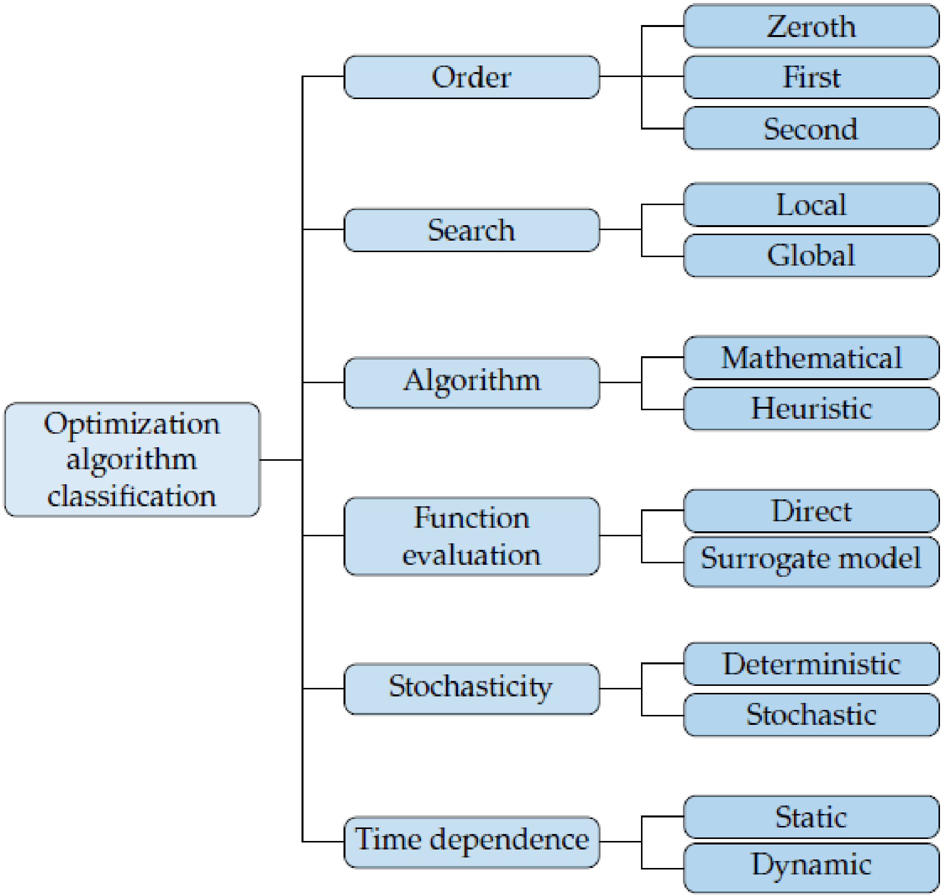


Figure 1.20 Classification of optimization algorithms based on attributes. These attributes are independent, and any combination is possible

Source: Martins et Ning, 2021

1.4.1. Order of information

- At the minimum, an optimization algorithm requires users to provide the models that compute the objective and constraint values – zeroth-order information – for any given set of allowed design variables.
- We call algorithms that use only these function values gradient-free algorithms (also known as derivative-free or zeroth-order algorithms).
- The advantage of gradient-free algorithms is that the optimization is easier to set up because they do not need additional computations other than what the models for the objective and constraints already provide.

1.4.1. Order of information

- Gradient-based algorithms use gradients of both the objective and constraint functions with respect to the design variables - first-order information.
- The gradients provide much richer information about the function behaviour, which the optimizer can use to converge to the optimum more efficiently.
- Figure 1.23 shows how the cost of gradient-based versus gradient-free optimization algorithms typically scales when the number of design variables increases.

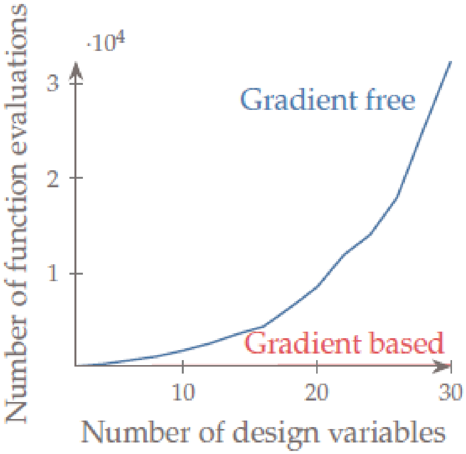


Figure 1.21 Gradient-based algorithms scale much better with the number of design variables.

Source: Martins et Ning, 2021

1.4.1. Order of information

- The number of function evaluations required by gradient-free methods increases dramatically, whereas the number of evaluations required by gradient-based methods does not increase as much and is many orders of magnitude lower for the larger numbers of design variables.
- In addition, gradient-based methods use more rigorous criteria for optimality.
- The gradients are used to establish whether the optimizer converges to a point that satisfies mathematical optimality conditions, something that is difficult to verify in a rigorous way without gradients.
- Gradient-based algorithms also include algorithms that use curvature – second-order information. Curvature is even richer information that tells us the rate of the change in the gradient, which provides an idea of where the function will flatten out.

1.4.1. Order of information

- Because gradient-based methods require accurate gradients and smooth enough functions, they require more knowledge about the models and optimization algorithm than gradient-free methods.

1.4.2. Local versus Global search

- The many ways to search the design space can be classified as being local or global.
- A local search takes a series of steps starting from a single point to form a trail of points that hopefully converges to a local optimum.
- In spite of the name, local methods can traverse large portions of the design space and can even step between convex regions (although this happens by chance).
- A global search tries to span the whole design space in the hopes of finding the global optimum.
- As previously mentioned, when discussing multimodality, even when using a global method, we cannot prove that any optimum found is a global one except for particular cases.

1.4.2. Local versus Global search

- The choice of search type is intrinsically linked to the modality of the design space.
- If the design space is unimodal, then a local search will be sufficient, and it will converge to the global optimum.
- If the design space is multimodal, a local search will converge to an optimum that might be local (or global if we are lucky enough).
- A global search will increase the likelihood that we converge to a global optimum, but this is by no means guaranteed.

1.4.3. Mathematical versus Heuristic

- There is a big divide regarding the extent to which an algorithm is based on provable mathematical principles.
- Optimization algorithms require an iterative process, which determines the sequence of points evaluated when searching for an optimum, and optimality criteria, which determine when the iterative process ends.
- Heuristics are rules of thumb or commonsense arguments that are not based on a strict mathematical rationale.
- Gradient-based algorithms are usually based on mathematical principles, both for the iterative process and for the optimality criteria.
- Gradient-free algorithms are more evenly split between the mathematical and heuristic for both the optimality criteria and the iterative procedure.

1.4.3. Mathematical versus Heuristic

- The mathematical gradient-free algorithms are often called derivative-free optimization algorithms.
- Heuristic gradient-free algorithms include a wide variety of nature-inspired algorithms.
- Heuristic optimality criteria are an issue because, strictly speaking, they do not prove a given point is a local (let alone global) optimum; they are only expected to find a point that is "close enough".
- This contrasts with mathematical optimality criteria, which are unambiguous about (local) optimality and converge to the optimum within the limits of the working precision.
- The mathematical criteria usually require the gradients of the objective and constraints.

1.4.3. Mathematical versus Heuristic

- This is not to suggest that heuristic methods are not useful.
- Finding a better solution is often desirable regardless of whether or not it is strictly optimal.
- Not converging tightly to optimality criteria does, however, make it harder to compare results from different methods.
- Most algorithms mix mathematical arguments and heuristics to some degree.
- Mathematical algorithms often include constants whose values end up being tuned based on experience.
- Conversely, algorithms primarily based on heuristics sometimes include steps with mathematical justification.

1.4.4. Function evaluation

- The optimization problem setup described previously assumes that the function evaluations are obtained by solving numerical models of the system.
- We call these direct function evaluations.
- However, it is possible to create surrogate models (also known as metamodels) of these models and use them in the optimization process.
- These surrogates can be interpolation-based or projection-based models.

1.4.5. Stochasticity

- This attribute is independent of the stochasticity of the model that we mentioned previously, and it is strictly related to whether the optimization algorithm itself contains steps that are determined at random or not.
- A deterministic optimization algorithm always evaluates the same points and converges to the same result, given the same initial conditions.
- In contrast, a stochastic optimization algorithm evaluates a different set of points if run multiple times from the same initial conditions, even if the models for the objective and constraints are deterministic.
- For example, most evolutionary algorithms include steps determined by generating random numbers.
- Gradient-based algorithms are usually deterministic, but some exceptions exist, such as stochastic gradient descent.

1.4.6. Time dependence

- Here, it assumed that the optimization problem is static.
- This means that the problem is formulated as a single optimization and the complete numerical model is solved at each optimization iteration.
- In contrast, dynamic optimization problems solve a sequence of optimization problems to make decisions at different time instances based on information that becomes available as time progresses.
- For some problems that involve time dependence, we can perform time integration to solve for the entire time history of the states and then compute the objective and constraint function values for an optimization iteration.
- This means that every optimization iteration requires solving for the entire time history.

1.4.6. Time dependence

- An example of this type of problem is a trajectory optimization problem where the design variables are the coordinates representing the path, and the objective is to minimize the total energy expended to get to a given destination.
- Although such a problem involves a time dependence, we still classify it as static because we solve a single optimization problem.
- As a more specific example, consider a car going around a racetrack. We could optimize the time history of the throttle, braking, and steering of a car to get a trajectory that minimizes the total time in a known racetrack for fixed conditions.
- This is an open-loop optimal control problem because the car control is predetermined and does not react to any disturbances.

1.5. Selecting an optimization approach

- It is useful to have a guidance on how to select an appropriate approach for solving a given optimization problem.
- This process cannot always be summarised into a simple decision tree; however, it is still helpful to have a framework as a first guide.
- Many of these decisions will become more apparent as one gains experience and, eventually, selecting an appropriate methodology will become second nature.
- Figure 1.22 outlines one approach to algorithm selection.
- The first two characteristics in the decision tree (convex problem and discrete variables) are not the most common within the broad spectrum of engineering optimization problems, but we list them first because they are the more restrictive in terms of usable optimization algorithms.

1.5. Selecting an optimization approach

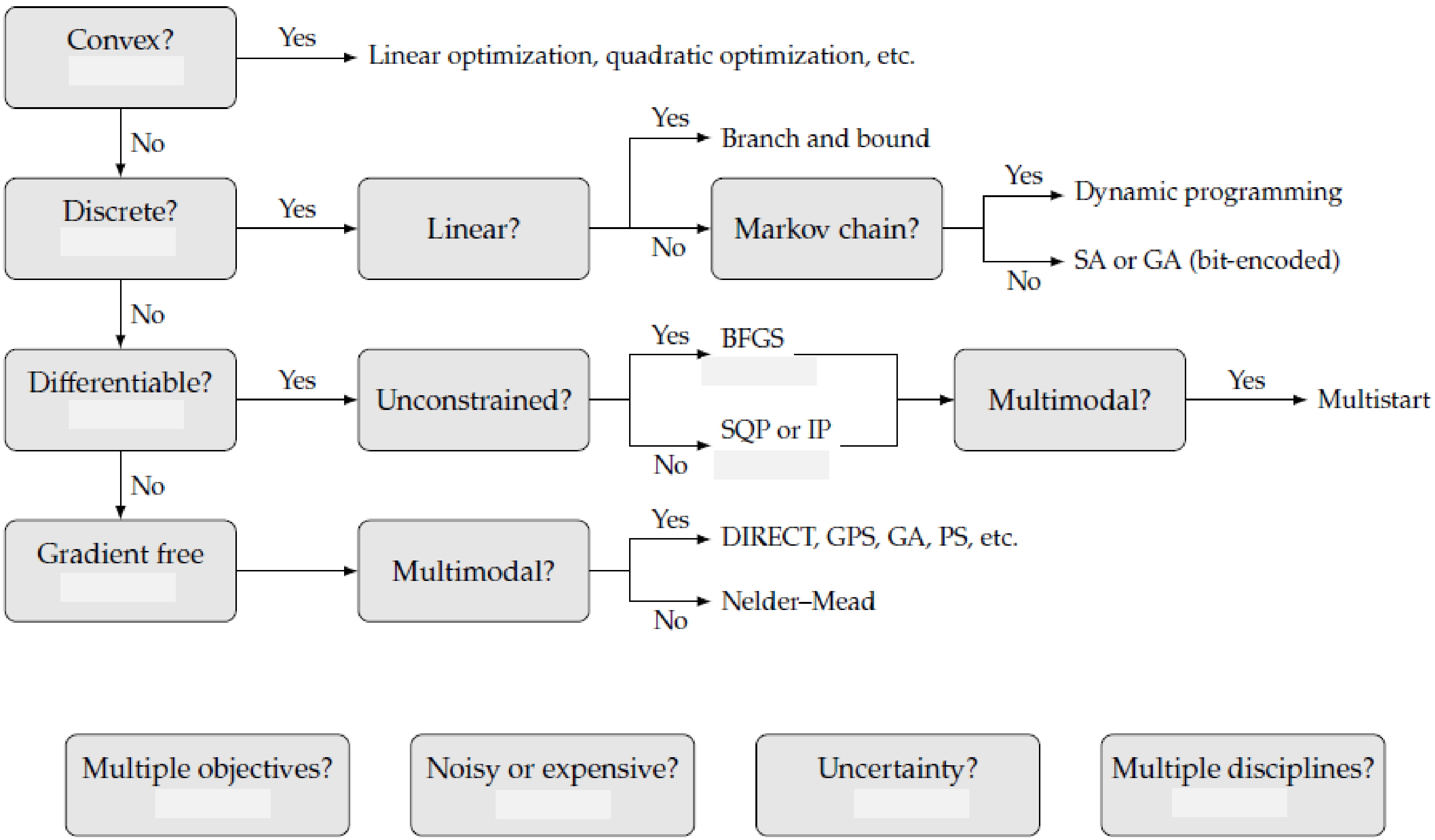


Figure 1.22 Decision tree for selecting optimization algorithms.

Source: Martins et Ning, 2021

1.5. Selecting an optimization approach

Convexity.

- Although it is often not immediately apparent if the problem is convex, with some experience, we can usually discern whether attempting to reformulate in a convex manner is likely to be possible.
- In most instances, convexity occurs for problems with simple objectives and constraints (e.g., linear or quadratic), such as in control applications where the optimization is performed repeatedly.
- A convex problem can be solved with general gradient-based or gradient-free algorithms, but it would be inefficient not to take advantage of the convex formulation structure if we can do so.

1.5. Selecting an optimization approach

Discrete variables.

- Problems with discrete design variables are generally much harder to solve, so we might consider alternatives that avoid using discrete variables when possible.
- For example, a wind turbine's position in a field could be posed as a discrete variable within a discrete set of options.
- Alternatively, we could represent the wind turbine's position as a continuous variable with two continuous coordinate variables.
- That level of flexibility may or may not be desirable but will generally lead to better solutions.
- Many problems are fundamentally discrete, and there is a wide variety of available methods, some of which can be very effective.

1.5. Selecting an optimization approach

Continuous and differentiable.

- If the problem is high dimensional (more than a few tens of variables as a rule of thumb), gradient-free algorithms are generally intractable and gradient-based algorithms are preferable.
- We would either need to make the model smooth enough to use a gradient-based algorithm or reduce the problem dimensionality to use a gradient-free algorithm.
- Another alternative if the problem is not readily differentiable is to consider surrogate-based optimization (the box labelled "Noisy or expensive" in Figure 1.22).
- If using the surrogate-based optimization route, we could still use a gradient-based approach to optimize the surrogate model because most such models are differentiable.

1.5. Selecting an optimization approach

- Finally, for problems with a relatively small number of design variables, gradient-free methods can be a good fit.
- Gradient-free methods have the largest variety of algorithms, and a combination of experience and testing is needed to determine an appropriate algorithm for the problem at hand.

1.5. Selecting an optimization approach

Algorithms.

- Linear optimization
- Quadratic optimization
- BnB (Branch and Bound) – is a method for solving optimization problems by breaking them down into smaller sub-problems and using a bounding function to eliminate sub-problems that cannot contain the optimal solution. It is an algorithm design paradigm for discrete and combinatorial optimization problems, as well as mathematical optimization.
- Dynamic programming
- SA (Simulated Annealing)
- GA (Genetic Algorithm)

1.5. Selecting an optimization approach

- BFGS – second-order optimization algorithm. It is an acronym, named for the four co-discovers of the algorithm: Broyden, Fletcher, Goldfarb, and Shanno. It is a local search algorithm, intended for convex optimization problems with a single optima.
- SQP (Sequential Quadratic Programming)
- IP (Integer Programming) – applicable to a special class of combinatorial optimization problems, which tend to be difficult to solve. It is a LP (Linear Programming) with integer variables.
- Multistart
- DIRECT - DIRECT global optimization algorithm initially provided an approach to minimizing a black-box function subject to lower and upper bounds on the variables. Many revisions allowed inclusion of constraints among other capabilities.

1.5. Selecting an optimization approach

- GPS (Generalized Pattern Search)
- PSO (Particle Swarm Optimization) – is a powerful meta-heuristic optimization algorithm and inspired by swarm behaviour observed in nature such as fish and bird schooling.
- Nelder-Mead – is a direct search method of optimization. It is used for non-differentiable objective functions and is generally referred to as a pattern search algorithm. The method works by evaluating a function at the vertices of a simplex, then iteratively shrinking the simplex as better points are found until some desired bound is obtained. Nelder-Mead repeatedly transforms the triangle of test points, replacing the worst point with a better one, and then contracts around a local minimum when it finds one.
- …

2. Examples of optimization problems

- Some practical optimization problems are presented next to illustrate the benefits of this approach to design practices.

2.1. Two-bar truss

- Consider the design of a simple tubular symmetric truss shown in Fig. 1.23 below (problem originally from Fox).
- A design of the truss is specified by a unique set of values for the analysis variables: height (H), diameter, (d), thickness (t), separation distance (B), modulus of elasticity (E), and material density (ρ).

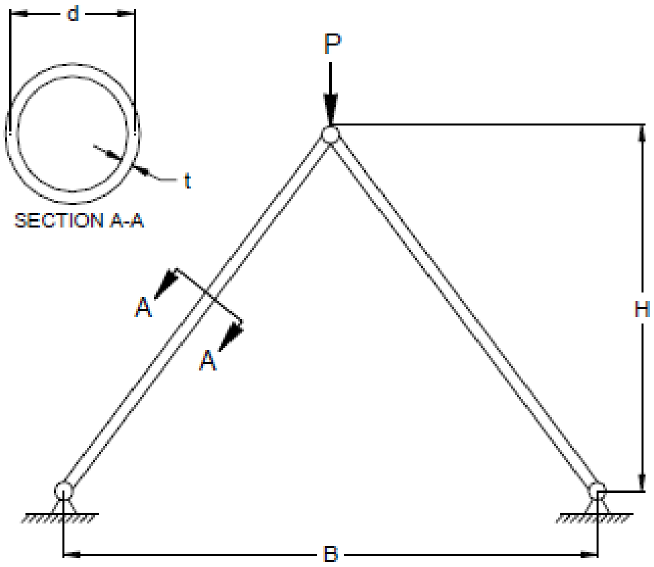


Figure 1.23 Layout for the Two-bar truss model.

Source: Parkinson et al, 2013

2.1. Two-bar truss

- Suppose we are interested in designing a truss that has a minimum weight, will not yield, will not buckle, and does not deflect "excessively", and so we decide our model should calculate weight, stress, buckling stress and deflection – these are the analysis functions.
- In this case we can develop a model of the truss using explicit mathematical equations. These equations are:

$$Weight = \rho \cdot 2 \cdot \pi \cdot d \cdot t \sqrt{\left(\frac{B}{2}\right)^2 + H^2}$$

$$Stress = \frac{P \cdot \sqrt{\left(\frac{B}{2}\right)^2 + H^2}}{2 \cdot t \cdot \pi \cdot d \cdot H}$$

$$Buckling\ Stress = \frac{\pi^2 E(d^2 + t^2)}{8[\left(\frac{B}{2}\right)^2 + H^2]}$$

$$Deflection = \frac{P \cdot \left[\left(\frac{B}{2}\right)^2 + H^2\right]^{(3/2)}}{2 \cdot t \cdot \pi \cdot d \cdot H^2 \cdot E}$$

2.1. Two-bar truss

- The analysis variables and analysis functions for the truss are also summarized in the table below.
- We note that the analysis variables represent all quantities on the right-hand side of the equations given above.
- When all of these are given specific values, we can evaluate the model, which refers to calculating the functions.

Analysis Variables	***Analysis Functions***
B, H, t, d, P, E, ρ	Weight, Stress, Buckling Stress, Deflection

2.1. Two-bar truss

- An example design for the truss is given below (left).
- We can obtain a new design for the truss by changing one or all analysis variable values.
- For example, if we change thickness from 0.15 in to 0.10 in., we find that weight has decreased, but stress and deflection have increased, as given below (right).

Analysis Variables	**Value**	**Value**
Height, H (in)	30.	30.
Diameter, d (in)	3.	3.
Thickness, t (in)	0.15	0.1
Separation distance, B (inches)	60.	60.
Modulus of elasticity (1000 lbs/in^2)	30,000	30,000
Density, ρ (lbs/in^3)	0.3	0.3
Load (1000 lbs)	66	66
Analysis Functions	**Value**	**Value**
Weight (lbs)	35.98	23.99
Stress (ksi)	33.01	49.52
Buckling stress (ksi)	185.5	185.3
Deflection (in)	0.066	0.099

2.1. Two-bar truss

- The optimization statement is

minimize	$Weight(x)$	
by varying	$x = [d, H]$	
subjected to	$Stress(x) \leq 100$	$i = 1,2$
	$(Stress - Buckling\ Stress)\ (x) \leq 0$	
	$Deflection(x) \leq 0.25$	

- Note that to define the buckling constraint, we have combined two analysis functions together.
- Thus we have mapped two analysis functions to become one design function.
- We can specify several optimization problems using the same analysis model.

2.1. Two-bar truss

- For example, we can define a different optimization problem for the two-bar truss to be

$$\begin{array}{lll} \text{minimize} & Stress(x) & \\ \text{by varying} & x = [t, d] & i = 1{,}2 \\ \text{subjected to} & Weight(x) \le 25 & \\ & Deflection(x) \le 0.25 & \end{array}$$

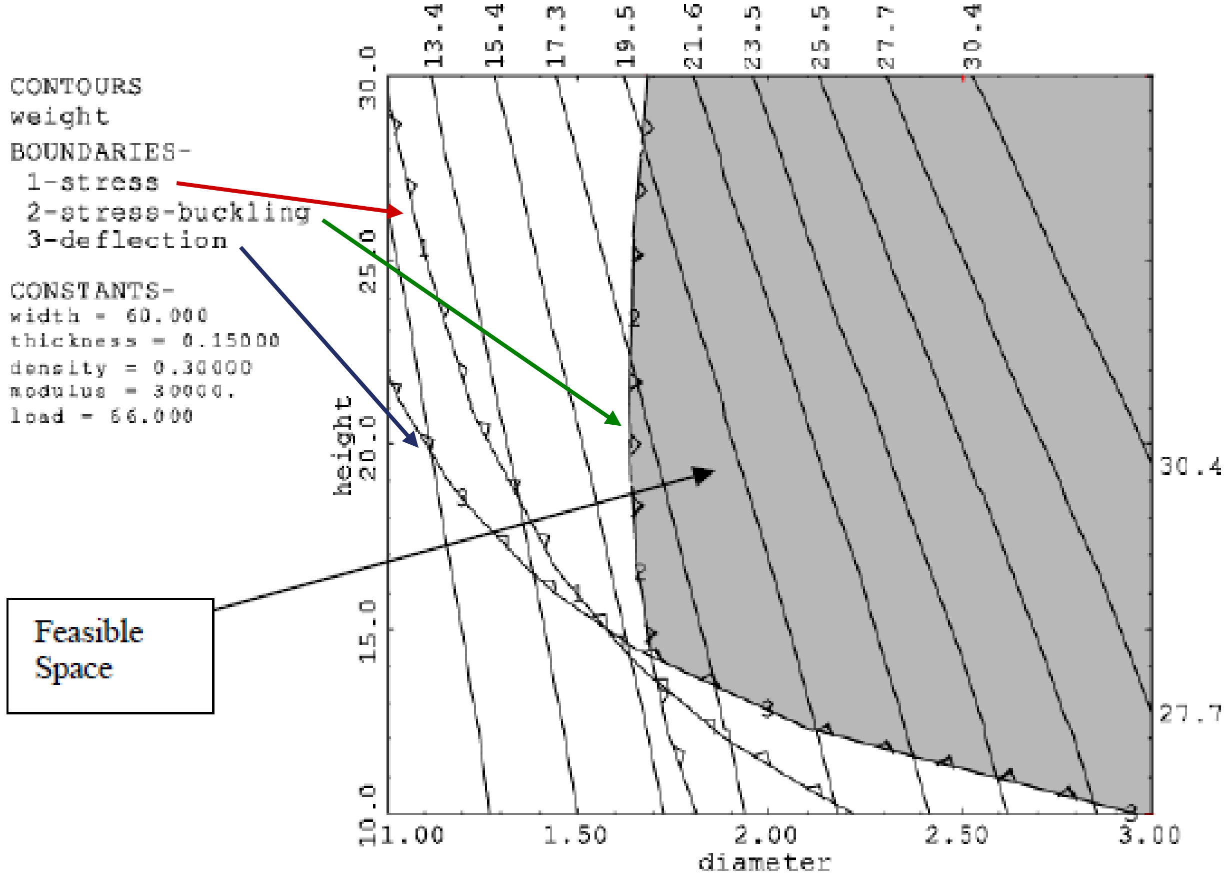
- The specifying of the optimization problem, i.e. the selection of the design variables and functions, is referred to as the mapping between the analysis space and the design space.
- For the problem defined in the previous slide, the mapping looks like

2.1. Two-bar truss

Analysis Space		**Design Space**
Analysis Variables		**Design Variables**
Height	→	Height
Diameter	→	Diameter
Thickness		
Width		
Density		
Modulus		
Load		
Analysis Functions		**Design Functions**
Weight	→	Minimize
Stress	→	Stress ≤ 100 ksi
Stress, Buckling Stress	→	(Stress – Buckling Stress) ≤ 0
Deflection	→	Deflection ≤ 0.25 inches

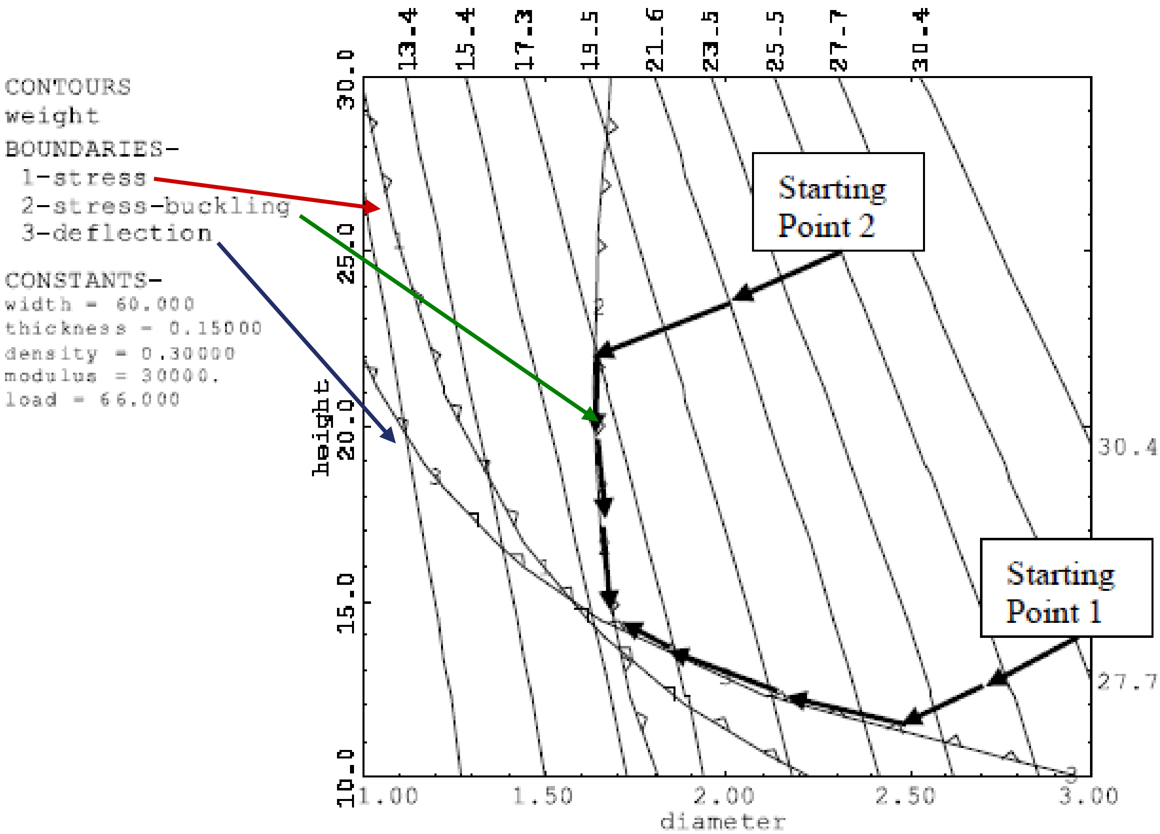
2.1. Two-bar truss

- Contour plot with design space:



2.1. Two-bar truss

- How an optimization algorithm works:



2.2. MDO for Aircraft Design at Bombardier Aerospace

- https://www.fzt.haw-hamburg.de/pers/Scholz/ewade/2015/SCAD2015_Piperni_MDOforAircraftDesignAtBombardier.pdf