CHAPTER 1

INTRODUCTION TO OPTIMIZATION-BASED DESIGN

1. What is Optimization?

Engineering is a profession whereby principles of nature are applied to build useful objects. A mechanical engineer designs a new engine, or a car suspension or a robot. A civil engineer designs a bridge or a building. A chemical engineer designs a distillation tower or a chemical process. An electrical engineer designs a computer or an integrated circuit.

For many reasons, not the least of which is the competitive marketplace, an engineer might not only be interested in a design which works at some sort of nominal level, but is the *best* design in some way. The process of determining the best design is called *optimization*. Thus we may wish to design the smallest heat exchanger that accomplishes the desired heat transfer, or we may wish to design the lowest-cost bridge for the site, or we may wish to maximize the load a robot can lift.

Often engineering optimization is done implicitly. Using a combination of judgment, experience, modeling, opinions of others, etc. the engineer makes design decisions which, he or she hopes, lead to an optimal design. Some engineers are very good at this. However, if there are many variables to be adjusted with several conflicting objectives and/or constraints, this type of experience-based optimization can fall short of identifying the optimum design. The interactions are too complex and the variables too numerous to intuitively determine the optimum design.

In this text we discuss a computer-based approach to design optimization. With this approach, we use the computer to search for the best design according to criteria that we specify. The computer's enormous processing power allows us to evaluate many more design combinations than we could do manually. Further, we employ sophisticated algorithms that enable the computer to efficiently search for the optimum. Often we start the algorithms from the best design we have based on experience and intuition. We can then see if any improvement can be made.

In order to employ this type of optimization, several qualifications must be met. First, we must have a *quantitative model* available to compute the responses of interest. If we wish to maximize heat transfer, we must be able to calculate heat transfer for different design configurations. If we wish to minimize cost, we must be able to calculate cost. Sometimes obtaining such quantitative models is not easy. *Obtaining a valid, accurate model of the design problem is the most important step in optimization*. It is not uncommon for 90% of the effort in optimizing a design to be spent on developing and validating the quantitative model. Once a good model is obtained, optimization results can often be realized quickly.

Fortunately, in engineering we often do have good, predictive models (or at least partial models) for a design problem. For example, we have models to predict the heat transfer or pressure drop in an exchanger. We have models to predict stresses and deflections in a bridge. Although engineering models are usually physical in nature (based on physical

principles), we can also use empirical models (based on the results of experiments). It is also perfectly acceptable for models to be solved numerically (using, for example, the finite element method).

Besides a model, we must have some variables which are free to be adjusted—whose values can be set, within reason, by the designer. We will refer to these variables as *design variables*. In the case of a heat exchanger, the design variables might be the number of tubes, number of shells, the tube diameters, tube lengths, etc. Sometimes we also refer to the number of design variables as the *degrees of freedom* of the computer model.

The freedom we have to change the design variables leads to the concept of *design space*. If we have four design variables, then we have a four dimensional design space we can search to find the best design. Although humans typically have difficulty comprehending spaces which are more than three dimensional, computers have no problem searching higher order spaces. In some cases, problems with thousands of variables have been solved.

Besides design variables, we must also have criteria we wish to optimize. These criteria take two forms: *objectives* and *constraints*. Objectives represent goals we wish to maximize or minimize. Constraints represent limits we must stay within, if inequality constraints, or, in the case of equality constraints, target values we must satisfy. Collectively we call the objectives and constraints *design functions*.

Once we have developed a good computer-based analysis model, we must link the model to optimization software. Optimization methods are somewhat generic in nature in that many methods work for wide variety of problems. After the connection has been made such that the optimization software can "talk" to the engineering model, we specify the set of design variables and objectives and constraints. Optimization can then begin; the optimization software will call the model many times (sometimes thousands of times) as it searches for an optimum design.

Usually we are not satisfied with just one optimum—rather we wish to explore the design space. We often do this by changing the set of design variables and design functions and re-optimizing to see how the design changes. Instead of minimizing weight, for example, with a constraint on stress, we may wish to minimize stress with a constraint on weight. By exploring in this fashion, we can gain insight into the trade-offs and interactions that govern the design problem.

In summary, *computer-based optimization* refers to using computer algorithms to search the design space of a computer model. The design variables are adjusted by an algorithm in order to achieve objectives and satisfy constraints. Many of these concepts will be explained in further detail in the following sections.

2. Engineering Models in Optimization

2.1. Analysis Variables and Functions

As mentioned, engineering models play a key role in engineering optimization. In this section we will discuss some further aspects of engineering models. We refer to engineering models as *analysis models.*

In a very general sense, analysis models can be viewed as shown in Fig 1.1 below. A model requires some inputs in order to make calculations. These inputs are called *analysis variables*. Analysis variables include design variables (the variables we can change) plus other quantities such as material properties, boundary conditions, etc. which typically would not be design variables. When all values for all the analysis variables have been set, the analysis model can be evaluated. The analysis model computes outputs called *analysis functions*. These functions represent what we need to determine the "goodness" of a design. For example, analysis functions might be stresses, deflections, cost, efficiency, heat transfer, pressure drop, etc. It is from the analysis functions that we will select the design functions, i.e., the objectives and constraints.

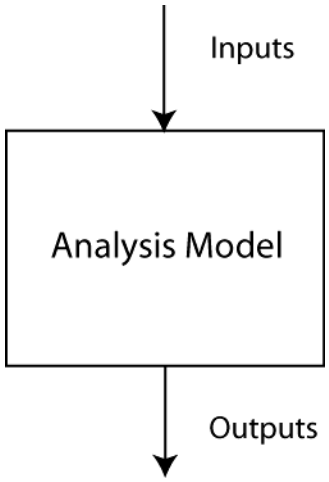


Fig. 1.1. The operation of analysis models

Thus from a very general viewpoint, analysis models require inputs—analysis variables—and compute outputs—analysis functions. Essentially all analysis models can be viewed this way.

2.2. An Example—the Two-bar Truss

At this point an example will be helpful.

Consider the design of a simple tubular symmetric truss shown in Fig. 1.2 below (problem originally from Fox[1]). 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

[1] R.L. Fox, *Optimization Methods in Engineering Design*, Addison Wesley, 1971

of elasticity (E), and material density (ρ). 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.

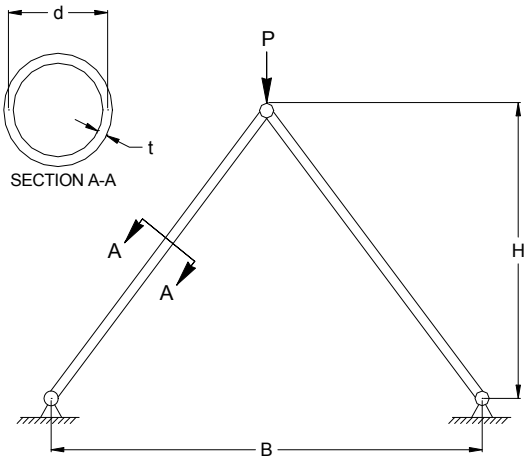


Fig. 1.2 - Layout for the Two-bar truss model.

In this case we can develop a model of the truss using explicit mathematical equations. These equations are:

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

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

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

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

The analysis variables and analysis functions for the truss are also summarized in Table 1.1. We note that the analysis variables represent all of the quantities on the right hand side of the

equations give above. When all of these are given specific values, we can *evaluate* the model, which refers to calculating the functions.

Table 1.1 - Analysis variables and analysis functions for the Two-bar truss.

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

An example design for the truss is given as,

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

We can obtain a new design for the truss by changing one or all of the 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,

Analysis Variables	**Value**
Height, H (in)	30.
Diameter, d (in)	3.
Thickness, t (in)	0.1
Separation distance, B (inches)	60.
Modulus of elasticity (1000 lbs/in^2)	30,000
Density, ρ (lbs/in^3)	0.3
Load (1000 lbs)	66
Analysis Functions	**Value**
Weight (lbs)	23.99
Stress (psi)	49.52

Buckling stress (psi)	185.3
Deflection (in)	0.099

3. Models and Optimization by Trial-and-Error

As discussed in the previous section, the "job," so to speak, of the analysis model is to compute the values of analysis functions. The designer specifies values for analysis variables, and the model computes the corresponding functions.

Note that the analysis software does not make any kind of "judgment" regarding the goodness of the design. If an engineer is designing a bridge, for example, and has software to predict stresses and deflections, the analysis software merely reports those values—it does not suggest how to change the bridge design to reduce stresses in a particular location. Determining how to improve the design is the job of the designer.

To improve the design, the designer will often use the model in an iterative fashion, as shown in Fig. 1.3 below. The designer specifies a set of inputs, evaluates the model, and examines the outputs. Suppose, in some respect, the outputs are not satisfactory. Using intuition and experience, the designer proposes a new set of inputs which he or she feels will result in a better set of outputs. The model is evaluated again. This process may be repeated many times.

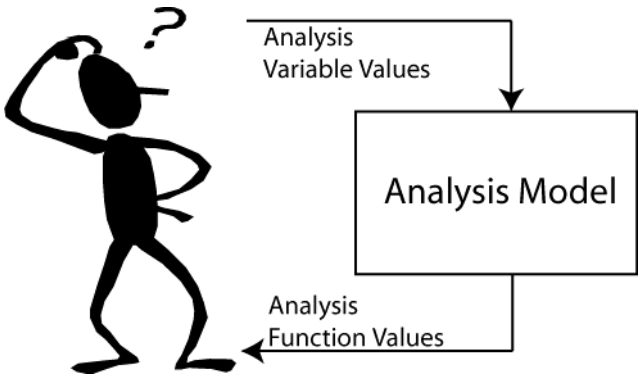


Fig. 1.3. Common "trial-and-error" iterative design process.

We refer to this process as "optimization by design trial-and-error." This is the way most analysis software is used. Often the design process ends when time and/or money run out.

Note the mismatch of technology in Fig 1.3. On the right hand side, the model may be evaluated with sophisticated software and the latest high-speed computers. On the left hand side, design decisions are made by trial-and-error. The analysis is high tech; the decision making is low tech.

4. Optimization with Computer Algorithms

Computer-based optimization is an attempt to bring some high-tech help to the decision making side of Fig. 1.3. With this approach, the designer is taken out of the trial-and-error

loop. The computer is now used to both evaluate the model and search for a better design. This process is illustrated in Fig. 1.4.

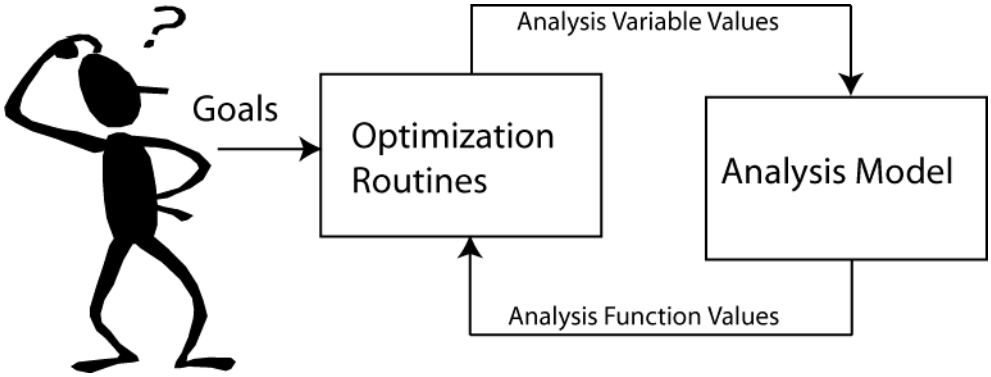


Fig. 1.4. Moving the designer out of the trial-and-error loop with computer-based optimization software.

The designer now operates at a higher level. Instead of adjusting variables and interpreting function values, the designer is specifying goals for the design problem and interpreting optimization results. The design space can be much more completely explored. Usually a better design can be found in a shorter time.

5. Specifying an Optimization Problem

5.1. Variables, Objectives, Constraints

Optimization problems are often specified using a particular form. That form is shown in Fig. 1.5. First the design variables are listed. Then the objectives are given, and finally the constraints are given. The abbreviation "s.t." stands for "subject to."

Find height and diameter to**:**

Minimize Weight

s. t.

Stress ≤ 100
(Stress-Buckling Stress) ≤ 0
Deflection ≤ 0.25

Fig. 1.5 Example specification of optimization problem for the Two-bar truss

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

Find thickness and diameter to:

Minimize Stress

s.t.

Weight ≤ 25.0
Deflection ≤ 0.25

Fig. 1.6 A second specification of an optimization problem for the Two-bar truss

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 Fig. 1.5 above, the mapping looks like:

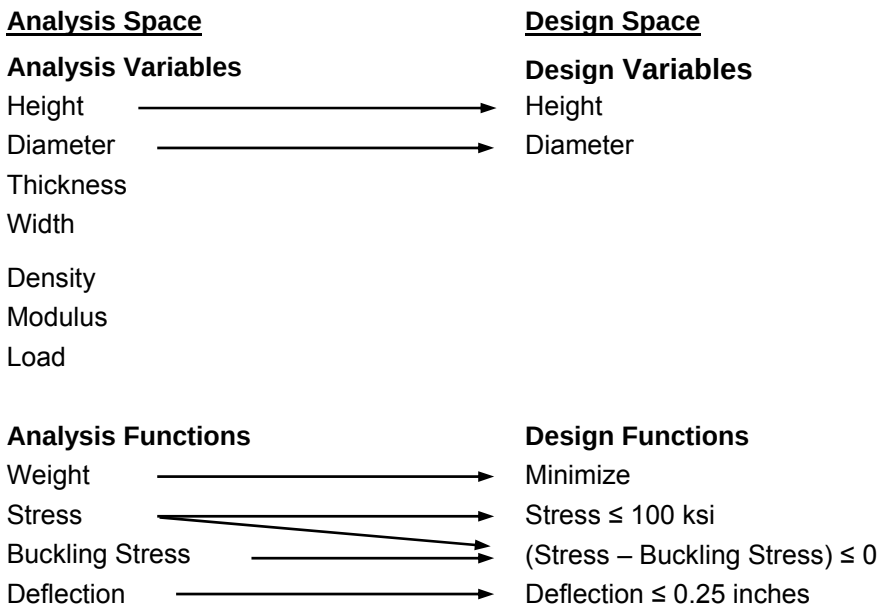


Fig. 1.7- Mapping Analysis Space to Design Space for the Two-bar Truss.

We see from Fig. 1.7 that the design variables are a subset of the analysis variables. This is always the case. In the truss example, it would never make sense to make load a design variable—otherwise the optimization software would just drive the load to zero, in which case the need for a truss disappears! Also, density and modulus would not be design variables unless the material we could use for the truss could change. In that case, the values these two variables could assume would be *linked* (the modulus for material A could only be matched with the density for material A) and would also be *discrete*. The solution of discrete variable optimization problems is discussed in Chapters 4 and 5. At this point, we will assume all design variables are continuous.

In like manner, the design functions are a subset of the analysis functions. In this example, all of the analysis functions appear in the design functions. However, sometimes analysis functions are computed which are helpful in understanding the design problem but which do

not become objectives or constraints. Note that the analysis function "stress" appears in two design functions. Thus it is possible for one analysis function to appear in two design functions.

In the examples given above, we only have one objective. This is the most common form of an optimization problem. It is possible to have multiple objectives. However, usually we are limited to a maximum of two or three objectives. This is because objectives usually are *conflicting* (one gets worse as another gets better) and if we specify too many, the algorithms can't move. The solution of multiple objective problems is discussed in more detail in Chapter 5.

We also note that all of the constraints in the example are "less-than" inequality constraints. The limits to the constraints are appropriately called the *allowable values* or *right hand sides*. For an optimum to be valid, all constraints must be satisfied. In this case, stress must be less than or equal to 100; stress minus buckling stress must be less than or equal to 0, and deflection must be less than or equal to 0.25.

Most engineering constraints are inequality constraints. Besides less-than ($\leq$) constraints, we can have greater-than ($\geq$) constraints. Any less-than constraint can be converted to a greater-than constraint or vice versa by multiplying both sides of the equation by -1. It is possible in a problem to have dozens, hundreds or even thousands of inequality constraints. When an inequality constraint is equal to its right hand side value, it is said to be *active* or *binding*.

Besides inequality constraints, we can also have equality constraints. Equality constraints specify that the function value should equal the right hand side. Because equality constraints severely restrict the design space (each equality constraint absorbs one degree of freedom), they should be used carefully.

5.2. Example: Specifying the Optimization Set-up of a Steam Condenser

It takes some experience to be able to take the description of a design problem and abstract out of it the underlying optimization problem. In this example, a design problem for a steam condenser is given. Can you identify the appropriate analysis/design variables? Can you identify the appropriate analysis/design functions?

Description:
Fig. 1.8 below shows a steam condenser. The designer needs to design a condenser that will cost a minimum amount and condense a specified amount of steam, m_{min}. Steam flow rate, m_S, steam condition, x, water temperature, T_W, water pressure P_W, and materials are specified. Variables under the designer's control include the outside diameter of the shell, D; tube wall thickness, t; length of tubes, L; number of passes, N; number of tubes per pass, n; water flow rate, m_W; baffle spacing, B, tube diameter, d. The model calculates the actual steam condensed, m_{cond}, the corrosion potential, CP, condenser pressure drop, ΔP_{cond}, cost, C_{cond}, and overall size, V_{cond}. The overall size must be less than V_{max} and the designer would like to limit overall pressure drop to be less than ΔP_{max}

Discussion:
This problem description is somewhat typical of many design problems, in that we have to infer some aspects of the problem. The wording, "Steam flow rate, m_s, steam condition, x, water temperature, T_w, water pressure P_w, and materials are specified," indicates these are either analysis variables that are not design variables ("unmapped analysis variables") or constraint right hand sides. The key words, "Variables under the designer's control…" indicate that what follows are design variables.

Statements such as "The model calculates…" and "The designer would also like to limit" indicate analysis or design functions. The phrase, "The overall size must be less than…" clearly indicates a constraint.

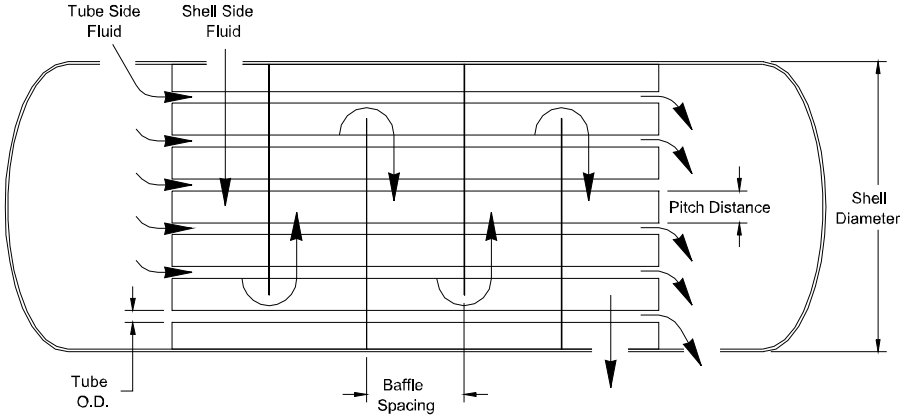


Fig. 1.8. Schematic of steam condenser.

Thus from this description it appears we have the following,

Analysis Variables:	**Design Variables:**
Outside diameter of the shell, D →	Outside diameter of the shell, D
Tube wall thickness, t →	Tube wall thickness, t
Length of tubes, L →	Length of tubes, L
Number of passes, N →	Number of passes, N
Number of tubes per pass, n →	Number of tubes per pass, n
Water flow rate, m_w →	Water flow rate, m_w
Baffle spacing, B, →	Baffle spacing, B,
Tube diameter, d →	Tube diameter, d.
Steam flow rate, m_s	
Steam condition, x	
Water temperature, T_w	
Water pressure, P_w	
Material properties	
Analysis Functions:	**Design Functions:**
Cost, C_{cond} →	Minimize Cost
Overall size, V_{cond} →	s.t.
Steam condensed, m_{cond} →	$V_{cond} \le V_{max}$
Condenser pressure drop, ΔP_{cond} →	$m_{cond} \ge m_{min}$
corrosion potential, CP →	$\Delta P_{cond} \le \Delta P_{max}$

6. Concepts of Design Space

So far we have discussed the role of analysis software within optimization. We have mentioned that the first step in optimizing a problem is getting a good analysis model. We have also discussed the second step: setting up the design problem. Once we have defined the optimization problem, we are ready to start searching the design space. In this section we will discuss some concepts of design space. We will do this in terms of the familiar truss example.

For our truss problem, we defined the design variables to be height and diameter. In Fig. 1.9 we show a contour plot of the design space for these two variables. The plot is based on data created by meshing the space with a grid of values for height and diameter; other analysis variables were fixed at values shown in the legend on the left. The solid lines (without the triangular markers) represent contours of weight, the objective. We can see that many combinations of height and diameter result in the same weight. We can also see that weight decreases as we move from the upper right corner of the design space towards the lower left corner.

Constraint boundaries are also shown on the plot, marked 1-3. These boundaries are also contour lines—the constraint boundary is the contour of the function which is equal to the allowable value. The small triangular markers show the *feasible* side of the constraint. The space where all constraints are feasible is called the *feasible space*. The feasible space for this example is shown in Fig. 1.9. By definition, an optimum lies in the feasible space.

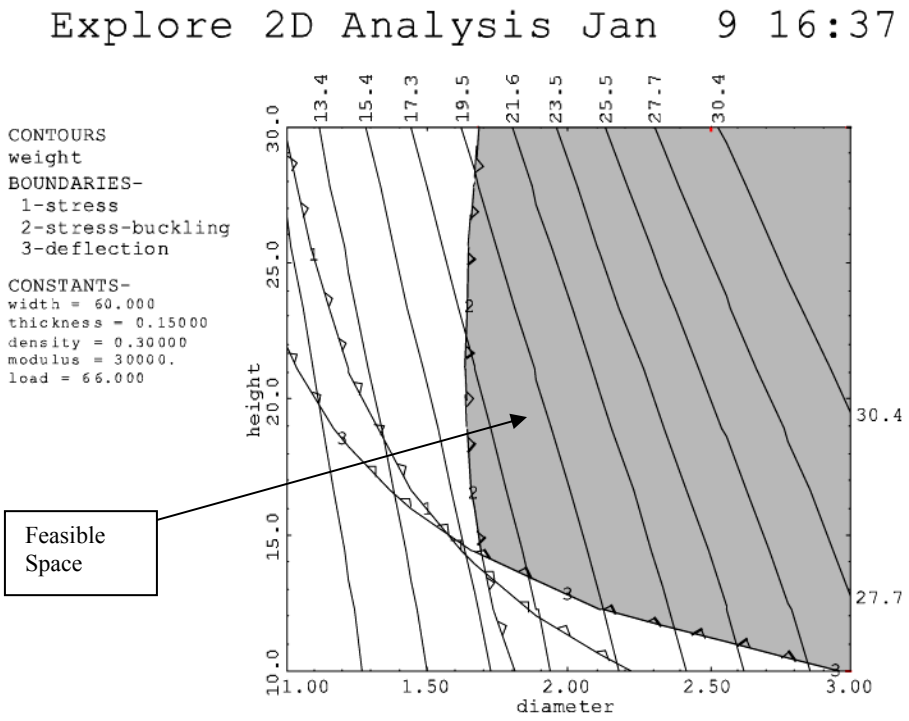


Fig. 1.9. Contour plot showing the design space for the Two-bar Truss.

We can easily see from the contours and the constraint boundaries that the optimum to this problem lies at the intersection of the buckling constraint and deflection constraint. We say that these are binding (or active) constraints. Several other aspects of this optimum should be noted.

First, this is a *constrained* optimum, since it lies on the constraint boundaries of buckling and deflection. These constraints are *binding* or *active.* If we relaxed the allowable value for these constraints, the optimum would improve. If we relaxed either of these constraints enough, stress would become a binding constraint.

It is possible to have an *unconstrained* optimum when we are optimizing, although not for this problem. Assuming we are minimizing, the contours for such an optimum would look like a valley. Contour plots showing unconstrained optimums are given in Fig. 1.10

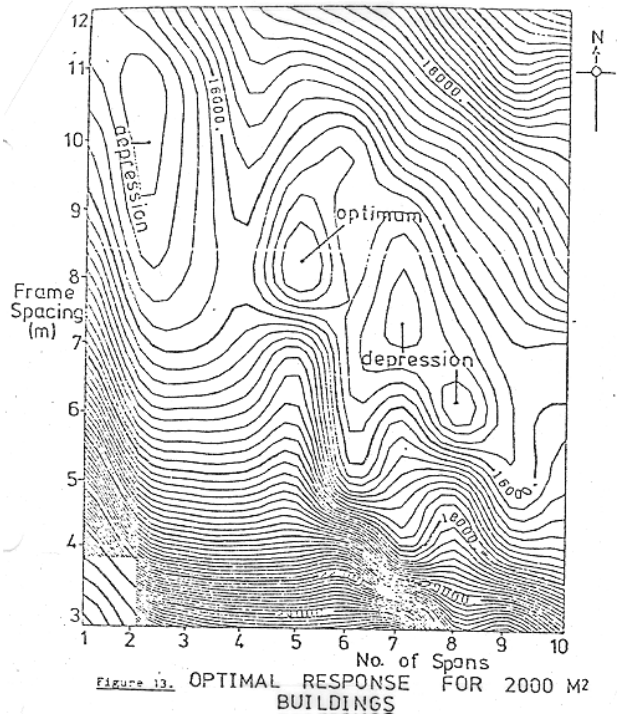
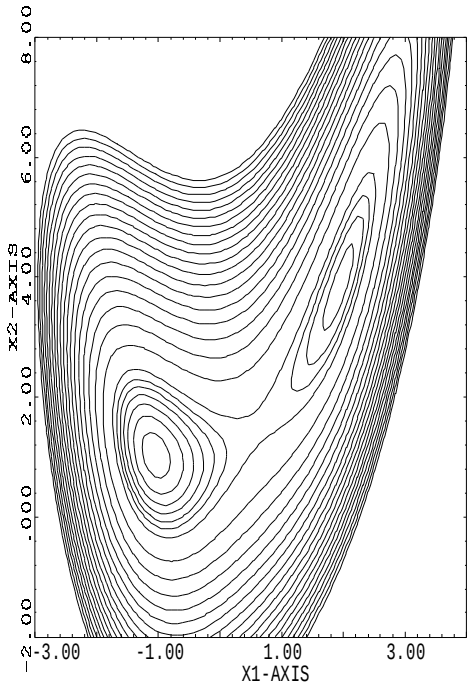
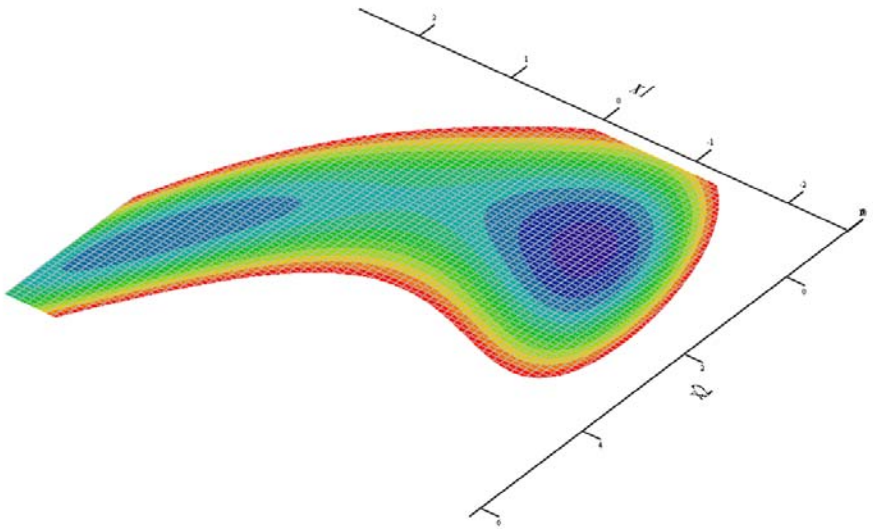


Figure 13. OPTIMAL RESPONSE FOR 2000 M² BUILDINGS



$$f(x_1,x_2) = 4 + 4.5x_1 - 4x_2 + x_1^2 + 2x_2^2 - 2x_1x_2 + x_1^4 - 2x_1^2x_2$$



$$f(x_1, x_2) = 4 + 4.5x_1 - 4x_2 + x_1^2 + 2x_2^2 - 2x_1x_2 + x_1^4 - 2x_1^2x_2$$

Fig. 1.10 Contour plots which show unconstrained optimums

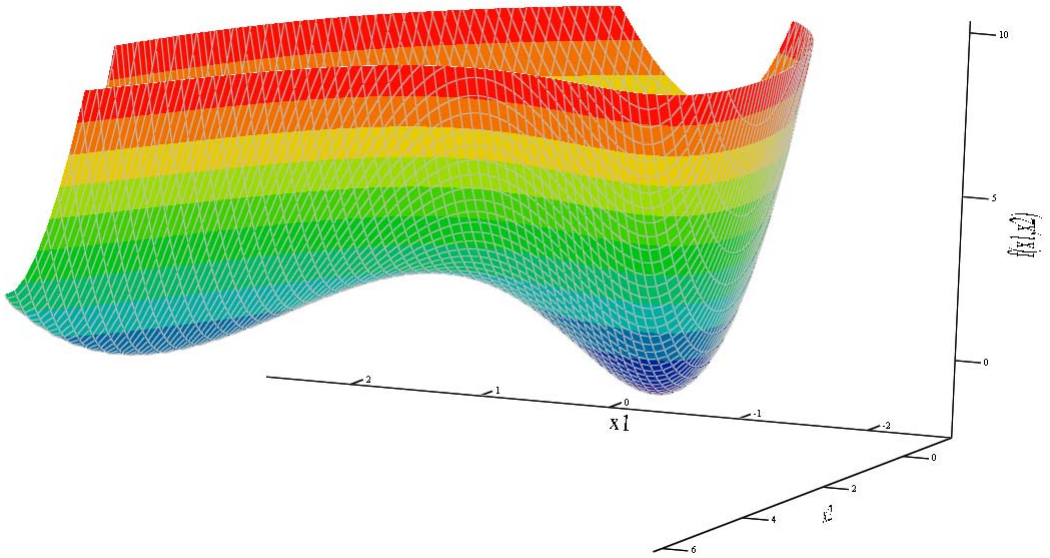


Fig. 1.11. Surface plot of the function:

$$f(x_1, x_2) := 4 + 4.5 \cdot x_1 - 4 \cdot x_2 + x_1^2 + 2 \cdot x_2^2 - 2 \cdot x_1 \cdot x_2 + x_1^4 - 2 \cdot x_1^2 \cdot x_2$$

Second, this optimum is a *local* optimum. A local optimum is a point which has the best objective value of any feasible point in its neighborhood. We can see that no other near-by feasible point has a better value than this optimum.

Finally, this is also a *global* optimum. A global optimum is the best value over the entire design space. Obviously, a global optimum is also a local optimum. However, if we have several local optimums, the global optimum is the best of these.

It is not always easy to tell whether we have a global optimum. Most optimization algorithms only guarantee finding a local optimum. In the example of the truss we can tell the optimum is global because, with only two variables, we can graph the entire space, and we can see that this is the only optimum in this space. If we have more than two variables, however, it is difficult to graph the design space (and even if we could, it might be too expensive to do so) so we don't always know for certain that a particular optimum is global. Often, to try to determine if a particular solution is a global optimum, we start the optimization algorithms from several different design points. If the algorithms always converge to the same spot, we have some confidence that point is a global optimum.

7. How Algorithms Work

The most powerful optimization algorithms for nonlinear continuous problems use derivatives to determine a search direction in *n* dimensional space that improves the objective and satisfies the constraints. Usually these derivatives are obtained numerically. For the Two-bar truss, search paths for two different starting points are given in Fig. 1.11

Assuming derivatives are obtained numerically, the number of times the algorithms must evaluate the model is roughly 10-15 times the number of optimization variables. Thus a problem with 3 variables will require roughly 30-45 evaluations of the model; a problem with 10 variables will require 100-150 model evaluations.

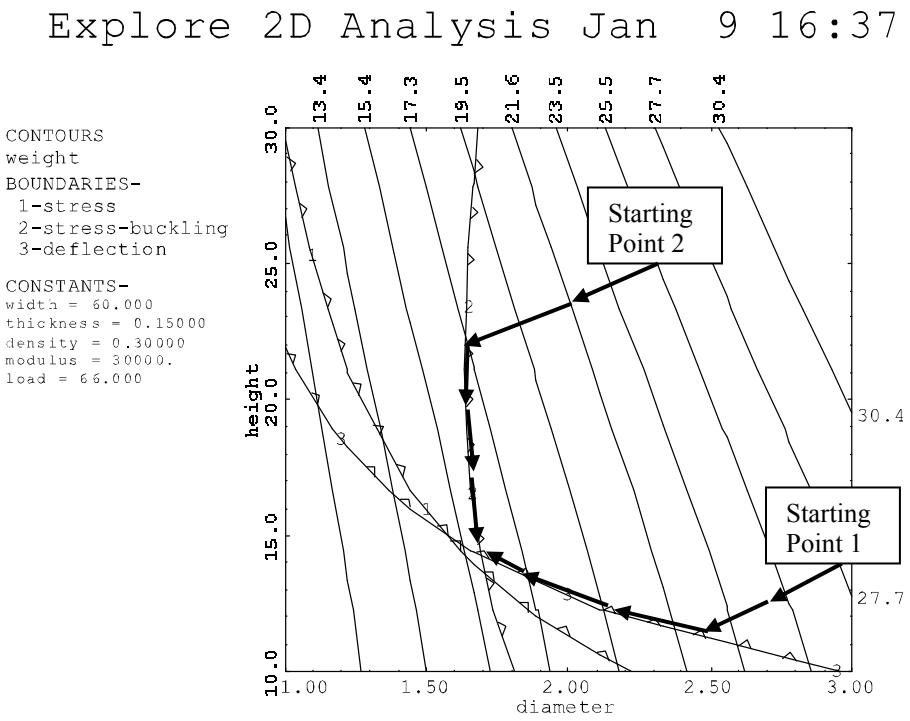


Fig. 1.12. Two paths algorithms might follow for two different starting points.

8. Cautions Regarding Optimization

Optimization algorithms can powerfully assist the designer in quantitative design synthesis. However,

1. The designer should always carefully and thoroughly validate the engineering model. Optimization of an inaccurate model is modestly illuminating at best and misleading and a waste of time at worst. Often optimization algorithms will exploit weaknesses in the model if they exist. As an example, regarding thermal systems, you can get some very high performance designs if a model does not enforce the second law of thermodynamics!

2. The algorithms help a designer optimize a particular design concept. At no point will the algorithms suggest that a different concept would be more appropriate. Achieving an overall optimum is dependent upon selection of the best concept as well as quantitative optimization.

3. Most engineering designs represent a compromise among conflicting objectives. Usually the designer will want to explore a number of alternative problem definitions to obtain insight and understanding into the design space. Sometimes non-quantitative considerations will drive the design in important ways.

4. We need to make sure the optimization problem represents the real problem we need to solve. Fox makes an important statement along these lines,

   > "In engineering design we often feel that the objective functions are self evident. But there are pitfalls. Care must be taken to optimize with respect to the objective function which most nearly reflects the *true* goals of the design problem. Some examples of common errors....should help focus this point. In static structures, the fully utilized (fully stressed) design is not always the lightest weight; the lightest weight design is not always the cheapest; in mechanisms, the design with optimum transmission angle does not necessarily have the lowest force levels; minimization of acceleration level at the expense of jerk (third derivative) may result in inadequate dynamic response."

5. We need to be careful to optimize the *system* and not just individual parts. As an example, the following poem by Oliver Wendell Holmes describes a "one horse" shay (a wagon) so well built that it lasted 100 years, after which it went completely to pieces all at once.

   But the Deacon swore (as Deacons do),
   With an "I dew vum," or an "I tell *yeou*,"
   He would build one shay to beat the taown
   'N' the keountry 'n' all the kentry raoun';
   It should be so built that it *couldn't* break daown:
   --"Fur," said the Deacon, "'t's mighty plain
   Thut the weakes' place mus' stan' the strain;
   'N' the way t' fix it, uz I maintain,
   Is only jest

> T' make that place uz strong uz the rest."
> ....
> You see, of course, if you're not a dunce,
> How it went to pieces all at once,--
> All at once, and nothing first,--
> Just as bubbles do when they burst.
> End of the wonderful one-hoss shay.
> Logic is logic. That's all I say.

Almost always in optimization we must take a system view, rather than an individual component view.