

Chapter 3

ACTUAL STATE IN OPTIMIZATION TECHNIQUES

Generally, the problem of optimal design of structures is formulated as follows:

Find the maximum or minimum of an objective (or cost) function subject to some constraints.

The classical methods used for such an optimization are, usually limited by the fact that the space of research is very large, the constraints are non-convex functions, and the gradients of these functions are not practically explicit. Overall, these methods will imply very long and expensive computations.

Special optimization algorithms were developed recently and are gaining popularity. In the following sections, we will review some of these algorithms, starting with some elementary notions underlining the difficulties in the optimal design of inelastic materials or structures.

3.1 EXTREMA OF A REAL FUNCTION

3.1.1 Free and Linked Extrema

Notations:

$f(x)$ is a real function of the real variable x in the interval (a, b) ; its derivative is denoted by $f'(x)$.

$f(x_1, \dots, x_n) = f(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^n$ is a scalar function of the n real variables x_1, \dots, x_n .

When $f(x_1, x_2, \dots, x_n)$ is a continuous function of all the independent variables (x_1, x_2, \dots, x_n) , there is a maximum or a minimum, *free extremum*, for the points in which the increment of the function is given by:

$\Delta f = f(\mathbf{x} + \Delta \mathbf{x}) - f(\mathbf{x}) \geq 0$ for any small variation of the variables around these points $\Delta \mathbf{x}$.

When the function is also derivable, it implies at least

$$\frac{\partial f}{\partial x_i}(x_1, x_2, \dots, x_n) = 0 \quad \forall i = 1, \dots, n.$$

When the n variables (x_1, \dots, x_n) are not independent but satisfy m_E conditions ($m_E \leq n$) (or equalities constraints):

$$c_1(x_1, \dots, x_n) = 0$$

⋮

$$c_{m_E}(x_1, \dots, x_n) = 0$$

the search of the extrema of $f(x_1, \dots, x_n)$ is more difficult. These extrema are called the *linked extrema*.

In this case, we still have to search for the points which satisfy

$\Delta f = f(\mathbf{x}^* + \Delta \mathbf{x}) - f(\mathbf{x}^*) \geq 0$, but now for any small *compatible* variation of the variables $\Delta \mathbf{x}$.

i) It is possible to eliminate m_E variables from the relations, that is to say to select $(n - m_E)$ independent variables

ii) Lagrange Multipliers

When we define the new function of $n + m_E$ variables (Lagrangian) :

$$\mathcal{L}(x_1, \dots, x_n, \mu_1, \dots, \mu_{m_E}) = f(x_1, \dots, x_n) + \sum_{j=1}^{m_E} \mu_j c_j(x_1, \dots, x_n)$$

and when we look at the free extrema of the new function for which

$$\Delta \mathcal{L} = \sum_i^n \frac{\partial \mathcal{L}}{\partial x_i} \Delta x_i + \sum_j^{m_E} \frac{\partial \mathcal{L}}{\partial \mu_j} \Delta \mu_j = 0, \forall \Delta x_i, \Delta \mu_j \text{ or else:}$$

$$\frac{\partial \mathcal{L}}{\partial x_i} = \frac{\partial f}{\partial x_i} + \sum_j^{m_E} \mu_j \frac{\partial c_j}{\partial x_i} = 0, \forall i = 1, n$$

$$\frac{\partial \mathcal{L}}{\partial \mu_j} = c_j = 0, \forall j = 1, m_E$$

3.1.2 Constrained Extrema

$f(\mathbf{x}) = f(x_1, \dots, x_n)$, is often denoted by the *cost* or *objective* function.

The n variables satisfy the m conditions, where the m_E conditions are equality conditions (simple conditions)

$$c_j(\mathbf{x}) = c_j(x_1, \dots, x_n) = 0, j = 1, m_E \quad (m_E \leq n)$$

and the $m - m_E$ conditions are inequality conditions (unilateral constraints)

$$c_j(\mathbf{x}) = c_j(x_1, \dots, x_n) \leq 0, j = m_E + 1, m$$

The great difficulty lies in the fact that we do not know a priori if the solution

$$\mathbf{x}^{T*} = \begin{pmatrix} x_1^* \\ \vdots \\ x_n^* \end{pmatrix}, \text{ for which } f(\mathbf{x}^*) \text{ is extrema, will satisfy the strict inequality}$$

$c_j(\mathbf{x}^*) < 0$, or on the contrary, the equality condition $c_j(\mathbf{x}^*) = 0$ for each $j = m_E + 1, m$. This solution is called a constrained extremum.

In the case of search for the minimum (or maximum) of a convex function $f(\mathbf{x})$ (or concave) and when all conditions $c_j(\mathbf{x}) \leq 0$ ($j = 1, m$) are also convex (or concave) functions, it is possible to demonstrate the necessary conditions to obtain the solution. The problem is called *convex optimisation*, and is then expressed by

$$\min f(\mathbf{x}) \text{ with } \mathbf{c}(\mathbf{x}) \leq 0$$

One \mathbf{x} which satisfy all the constraints $\mathbf{c}(\mathbf{x}) \leq 0$ is called a *feasible* point. As all the constraints are convex functions, the set of feasible points is a convex set C . There is a solution only if this set a non void set.

As the linked extrema problem, the Generalized Lagrangian defined by:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{x}) = f(\mathbf{x}) + \sum_{j=1}^m \lambda_j c_j(\mathbf{x})$$

function of $n + m$ variables,

$$\mathbf{x}^T = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad \boldsymbol{\lambda}^T = \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_m \end{pmatrix} \quad \text{with all } \lambda_j \geq 0.$$

The λ_j are denoted the generalized Lagrange multipliers. This function $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ is (for a

given λ) a convex function of \mathbf{x} and a linear (then concave) function of λ (for a given \mathbf{x}). It is then possible to demonstrate that \mathbf{x}^* is a solution if there exists one $\lambda^* \geq 0$ such that for all \mathbf{x} and $\lambda \geq 0$, we have the Kuhn-Tucker conditions:

$$\mathcal{L}(\mathbf{x}^*, \lambda) \leq \mathcal{L}(\mathbf{x}^*, \lambda^*) \leq \mathcal{L}(\mathbf{x}, \lambda^*)$$

3.1.3 Particular cases

- When the cost function $f(\mathbf{x})$ as all the constraints $\mathbf{c}(\mathbf{x})$ are linear function of \mathbf{x} , find the minimum of $f(\mathbf{x}) = \mathbf{b}^T \mathbf{x} + e$ with $\mathbf{c}(\mathbf{x}) = \mathbf{A} \mathbf{x} + d \leq 0$

$f(\mathbf{x}) = \sum_{i=1}^n b_i x_i + e$ with the conditions $c_j(x) = \sum_{i=1}^n A_{ji} x_i + d_j \leq 0$ the simplex algorithm is used. This algorithm is based on the property that the solution is necessary a corner of the set C limited by the hyperplanes $\mathbf{c}(\mathbf{x}) \leq 0$.

- When the cost function $f(\mathbf{x})$ is a quadratic and all the constraints are linear functions of \mathbf{x} find the minimum of $f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{C} \mathbf{x} + d^T \mathbf{x} + e$ with $\mathbf{c}(\mathbf{x}) = \mathbf{A} \mathbf{x} + \mathbf{b} \leq 0$

\mathbf{C} and \mathbf{A} being positive symmetrical matrices.

The generalized Lagrangian a the form:

$$\mathcal{L}(\mathbf{x}, \lambda) = \frac{1}{2} \mathbf{x}^T \mathbf{C} \mathbf{x} + d^T \mathbf{x} + \lambda^T (\mathbf{A} \mathbf{x} + \mathbf{b}), \quad \lambda \geq 0$$

and the Kuhn-Tucker conditions are expressed by:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \mathbf{C} \mathbf{x} + d + \mathbf{A}^T \lambda = 0$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \mathbf{A} \mathbf{x} + \mathbf{b} \leq 0$$

$$\lambda^T (\mathbf{A} \mathbf{x} + \mathbf{b}) = 0$$

$$\lambda \geq 0$$

3.2 MATHEMATICAL PROGRAMMING

Several algorithms were designed to solve the general problem in which the cost function and/or the constraints functions are non-convex functions. Of course, for such a problem, the uniqueness of the solution is no more insured in general. We have to find the minimum (or the maximum) of $f(\mathbf{x})$ of the n variables $\mathbf{x} = (x_1, \dots, x_n)$

$$\text{which satisfy } \mathbf{x} \in \mathfrak{R}^n \text{ and } \begin{array}{ll} (a). & c_i(\mathbf{x}) = 0, \quad i = 1, \dots, m_E \\ (b). & c_i(\mathbf{x}) \leq 0, \quad i = m_E + 1, \dots, m \\ (c). & x_j^l \leq x_j \leq x_j^u, \quad j = 1, \dots, n \end{array}$$

These conditions define in \mathfrak{R}^n the *feasible domain*. As in the convex programming, the algorithms are such that they have to reduce the cost function and in the same time keep the point in the feasible domain at any time.

Most of the algorithms belong to the two classes:

- the gradient based methods which imply the computations of the gradients of all functions
- the only values of the functions methods.

The strategies are then deterministic or random selections of the points in the feasible space.

3.2.1 Gradient methods

These methods require that the gradients of the functions are computed.

$\nabla_x h$ denotes the first derivative of the function h relative to \mathbf{x} or its gradient

$$\nabla_x h(\mathbf{x}) = \left(\frac{\partial}{\partial x_1} h(\mathbf{x}), \dots, \frac{\partial}{\partial x_n} h(\mathbf{x}) \right)^T$$

and $\nabla_x^2 h$ denotes the second derivative of the function h relative to \mathbf{x} or its Hessian

$$\nabla_x^2 h(\mathbf{x}) = \left(\frac{\partial^2}{\partial x_i \partial x_j} h(\mathbf{x}) \right)_{i,j=1,n}$$

$D_x \mathbf{H}(\mathbf{x})$ denotes the Jacobian of \mathbf{H} (m functions of n or an application of $\mathfrak{R}^n \rightarrow \mathfrak{R}^m$):

$$\mathbf{H}(\mathbf{x}) = (H_1(\mathbf{x}), \dots, H_m(\mathbf{x}))^T,$$

$$D_x \mathbf{H}(\mathbf{x}) = \left(\frac{\partial}{\partial x_j} H_i(\mathbf{x}) \right)_{i=1,m; j=1,n}$$

The generalized Lagrangian is defined by

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i \in \mathcal{E}} \mu_i c_i(\mathbf{x}) + \sum_{i \in \mathcal{I}} \lambda_i c_i(\mathbf{x})$$

For one point in the feasible domain, the unilateral constraints are divided into two groups according to the fact that they may *active* if $c_i(\mathbf{x}^*) = 0$, $i \in \mathcal{I}$ or *inactive* otherwise.

The optimization algorithms are iterative processes, they will start by a first estimation of a feasible point \mathbf{x}^0 . Then the point is typically modified as:

$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha d^k$, where k is the actual index of the iteration, the vector d^k is the search direction in \mathfrak{R}^n , usually the gradient or the conjugated gradient, and the scalar α is the step of the motion in the direction.

There are two main steps: it is necessary to find a good direction and then in this direction, the good step.

Several algorithms were proposed. Most of them are based on the Modified Newton

method such as in the formulation by Powell (RQP Recursive Quadratic Programming). Other ones used the Generalized Reduced Gradient (GRG).

At last, some methods transform the problem is a sequence of problems without any constraints as in Sequential Methods (*SUMT*).

3.2.2 Stochastic Evolutionary Algorithms: Genetic Algorithms

In the feasible domain, some points are "cleverly randomly" chosen and their performance is evaluated. Very often these algorithms are mixed with one gradient algorithms and they allow not to be locked in a local minimum.

Among them, the genetic algorithms (J. Holland) have been intensively applied for problems where the derivative of the functions can be computed and where some of the variables are discrete variables (enumerated variables) and can take only some special values.

These algorithms always imply a great number of iterations. However, they are helped by the fact that a population of individuals of a certain species will, after many generations, adapt to its environment. In other words, the members of this population will tend to disappear if they cannot adjust, but will live to reproduce if they adjust well enough; the good parents will give better adjustable children. and some genetic characters of their parents will be carried in chromosomes of the individuals (Darwinian evolution). Considering these, the principle of the genetic algorithms is explained simply as follows:

At first, each point or individual x is codified, i.e., represented by one string or a list of *chromosomes* (this coding scheme is an art and has to be developed for each problem). In the chromosome, we have a well defined sequence of *genes*. The coding has to allow the representation of continuous or discrete variable as one item in one list or one exclusive item in one list.

A *continuous* chromosome, represented according to the required accuracy by either 8, 16, or 32 genes, which are implemented in the computer as binary bits. (the distinct values of a gene, 0 and 1, are called alleles). *Enumerated* chromosomes are used in on combinatorial problem. *Multiple* chromosomes make up the individual.

So, each point has an internal representation within the algorithm, but it corresponds to one point in the real world at the end (decoding).

Usually the iterations are based on

- creation of one *population* of *individuals* represented by their *chromosomes*
- through a *fitness function*, evaluation of the *performance* of each individual
- generation of a new population by making the operations *selection*, proportional to their performance, *cross-over* and *mutation* on the individuals,
- substitution by the new population and loop until one criterium is reached. Each such a loop is called one *generation*.

At the beginning, the population is randomly generated; then some genetic operators will help improve the new population.

Section 3.2 MATHEMATICAL PROGRAMMING

cross-over 2 chromosomes or *parents*, are cut in the same position (randomly chosen) and they exchange their part to generate 2 new chromosomes or *children*.

mutation : randomly, a small perturbation is introduced in the chromosome.

reproduction : new individuals are created after cross-over and mutation.

selection it is necessary to favorize the individuals with the best performance.

We may note a very interesting and practical program based on this algorithm, GENEHUNTER from Wards Systems. Some examples of the applications of the program will be shown later.