MECH 577 Optimum Design

Lecture Notes

Jorge Angeles

Department of Mechanical Engineering &
Centre for Intelligent Machines
McGill University, Montreal (Quebec), Canada

These lecture notes are not as yet in final form.
Please report corrections & suggestions to

Prof. J. Angeles
Department of Mechanical Engineering &
McGill Centre for Intelligent Machines
McGill University
817 Sherbrooke St. W.
Montreal, Quebec
CANADA H3A 2K6
FAX: (514) 398-7348
angeles@cim.mcgill.ca

Contents

Notation

$\mathbf{1}$: The $n \times n$ identity matrix, when n is obvious

$\mathbf{1}_k$: the $k \times k$ identity matrix, when k should be specified

$\mathbf{A}$: $q \times n$ coefficient matrix of the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$

$\mathbf{A}^I$: the *left* Moore-Penrose generalized inverse (LMPGI) of the *full-rank* $q \times n$ matrix $\mathbf{A}$, with $q > n$:

$$\mathbf{A}^I \equiv (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T \tag{1}$$

$\mathbf{b}$: q-dimensional vector of the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$

$\mathbf{C}$: $p \times n$, with $p < n$, coefficient matrix of the *underdetermined* linear system $\mathbf{C}\mathbf{x} = \mathbf{d}$

$\mathbf{C}^\dagger$: the *right* Moore-Penrose generalized inverse (RMPGI) of the *full-rank* $p \times n$ matrix $\mathbf{C}$, with $p < n$:

$$\mathbf{C}^\dagger \equiv \mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1} \tag{2}$$

$\mathbf{d}$: p-dimensional vector of the linear system $\mathbf{C}\mathbf{x} = \mathbf{d}$

f: scalar objective function $f(\mathbf{x})$ to be minimized

$\mathbf{g}(\mathbf{x})$: p-dimensional nonlinear vector function of the set of inequalities $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$

$\mathbf{G}$: $p \times n$ Jacobian matrix of vector function $\mathbf{g}(\mathbf{x})$ w.r.t. $\mathbf{x}$

$\mathbf{H}$: the $n \times n$ Hessian matrix of the objective function $f(\mathbf{x})$

$\mathbf{H}_i$: ith Householder reflection used to render a rectangular matrix into upper-triangular form; a square matrix

$\mathbf{h}(\mathbf{x})$: l-dimensional nonlinear vector function of $\mathbf{x}$, occurring in the equality constraints $\mathbf{h}(\mathbf{x}) = \mathbf{0}$

$\mathbf{J}(\mathbf{x})$: $l \times n$ *gradient* of $\mathbf{h}$ w.r.t. $\mathbf{x}$

$\mathbf{L}$: lower-triangular matrix of the LU-decomposition of a square matrix $\mathbf{A}$. Also used to denote the *orthogonal complement* of $\mathbf{C}$ or $\mathbf{G}$; confusion is avoided because of the two different contexts in which these matrices occur

l: number of equality constraints $h_i(\mathbf{x}) = 0$, for $i = 1, \ldots, l$, expressed in vector form as $\mathbf{h}(\mathbf{x}) = \mathbf{0}$

m: number of equations $\phi_i(\mathbf{x}) = 0$, for $i = 1, \ldots, m$, expressed in vector form as $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{0}$

n: number of design variables x_i, for $i = 1, \ldots, n$, expressed in vector form as $\mathbf{x}$

$\mathbf{O}_{mn}$: the $m \times n$ zero matrix

p: number of constraint equations $g_i(x) = 0$, for $i = 1, \ldots, p$, expressed in vector form as $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ or $\mathbf{Cx} = \mathbf{d}$

q: number of equations in $\mathbf{Ax} = \mathbf{b}$

$\mathbf{U}$: (square) upper-triangular matrix

$\mathbf{V}$: $m \times m$ lower-triangular matrix, a factor of $\mathbf{W}$, i.e., $\mathbf{W} = \mathbf{V}^T\mathbf{V}$

$\mathbf{W}$: $m \times m$ symmetric and positive-semidefinite weighting matrix

$\mathbf{x}$: n-dimensional vector of design variables

$\mathbf{x}_0$: minimum-norm solution of an underdetermined linear system

$\mathbf{x}_L$: least-square solution of an overdetermined linear system

x_i: the ith component of vector $\mathbf{x}$

$\mathbf{x}^k$: the kth entry of a sequence $\mathbf{x}^0, \mathbf{x}^1, \ldots$

∇: the *gradient operator*, pronounced "nabla"; when its operand is a scalar, it yields a vector; when a vector, it yields a matrix

$\nabla\nabla$: the *Hessian operator*; its operand being a scalar, it produces a square, symmetric matrix

$\|\cdot\|$: *a* norm of either vector or matrix $(\cdot)$

Chapter 1

Preliminaries

1.1 The Role of Optimization Within the Design Process

The English word *design* derives from the Latin word *designare*, which means "to mark out"—as found, for example, in the *Random College Dictionary*. The word thus implies a goal, an objective. As such, the meaning of the word is extremely broad, encompassing the general activity of producing concepts aimed at a given goal, be this pure intellectual pleasure, in the realm of art, or pragmatic, in the realm of engineering.

The product of the design activity is a good, whether tangible, e.g., a fountain pen in the realm of *industrial design*, or intangible, e.g., a business plan, in the realm of *management*. We focus here on *engineering design*, but this does not mean that we exclude intangible goods. An important branch of engineering is *production systems*, whereby the design good is many a time intangible, such as the *organization* of a healthcare system.

Design is an extremely complex process. Various models have been proposed in the literature, e.g., the one by French (1992), which divides the process into two parts, (*i*) stages/descriptions and (*ii*) activities. The process is represented as a flow diagram, in which stages or descriptions are included in circles, while activities in rectangles. In this model, the process starts with a need and ends with "working drawings." In-between, we have a sequence, starting by the *analysis of the problem*, an activity, followed by the *statement of the problem*, a stage. Once the problem has been formulated, in design engineering terms, we suppose, as opposed to "client-needs" terms, the activity leading to the *conceptual design* follows, out of which comes (come) the "selected scheme(s)," apparently a description. Then comes the "embodiment of scheme(s)," i.e., of the scheme(s) selected in the previous part. The embodiment is then followed by the "detailing" of the designed object, also an activity.

However, the design process is recognized as being anything but a one-way street. Two feedback loops are thus included in French's model: One at the conceptual design level, in which a revision may indicate that the problem needs further analysis, and hence, the designer or the design team must return to the first activity. A second loop arises at the embodiment level, at which the designer may realize that either a revision of the conceptual design is needed, or even a revision of the problem analysis is warranted.

The foregoing model contemplates the design process as a consultant's activity, in which a consultant—an individual or a company—participates in a project within an organization, to either develop a new product or improve an existing one. The model does not contemplate the prototyping aspects—usually outside of the scope of the consultant's activities—that lead to the realization of the design motivated by the client's need. Prototyping is needed when either an innovative product is under design or when an improvement on an existing design is planned that will affect thousands or millions of produced objects. Prototype tests may bring about various feedback loops in turn. We propose in Fig. 1.1 an alternative model that should help better understand the role of optimization within the design process and its place therein.

In the model of Fig. 1.1, the design process is initiated by a client's need. The designer then translates the need into a form that allows the designer to analyze the client's needs within a design context, which is the block indicated as "Problem Definition." Once the problem is well-defined, free of the fuzziness of the client's description, a search for alternative solutions begins, leading to a design candidate. These activities take place within the "Preliminary Design" box in the proposed model. Once a design candidate has been selected, a detailed design follows, as included in the dashed box of the same model. In this phase, the input is a preliminary design solution, devoid of details. That is, the preliminary design is nothing but a rough layout needing an *embodiment*, i.e., a detailed definition in terms of materials, dimensions, and so on. The first step in this stage is the *synthesis* of the embodiment, i.e., a *topological* layout of the design—number of moving parts; types of moving parts; individual functions of the parts; etc.—involving only a qualitative description of the design structure. Once a topological layout has been produced, its design features are identified and labelled, probably using mathematical symbols, such as ℓ for length; m for mass; R for resistor; P for compressive load; T for tensile load; etc.

A key step in the dashed box consists in assigning numerical values to the foregoing features, which is a task calling for discipline-specific knowledge—fluid mechanics; structural engineering; machine design; multibody dynamics; etc.—either theoretical or empirical. Out of this knowledge comes a mathematical model relating all the foregoing features. Numerical values can now be assigned to these features using most often good *engineering judgment*, which comes only from experience. As a means of verifying decisions on dimensioning, the designer can resort to well-developed design methods leading systematically to the best possible values of those features, while satisfying the client's needs and budgetary constraints. Under "budgetary" we understand not only financial resources, but also time, for deadlines must be respected. How to assign values systematically to the foregoing features is the role of *optimization*.

Optimization is thus a process by which the decision-maker, in our case the designer, arrives at *optimum* values of the features defining the design solution proposed. A set of optimum values has been achieved when a cost has been minimized or a profit has been maximized, while respecting the mathematical model, i.e., the *functional relations* among all quantities at stake and the budgetary constraints expressed in the form of equality or inequality relations. Once all design features have been determined, and validated by means of simulation using discipline-specific tools—computational fluid dynamics code; finite element code; electromagnetic design code; code implementing Monte Carlo methods; etc.—an embodiment of the design solution can be produced, probably as a virtual prototype. Such embodiment is the output of the dashed box.

The embodiment is then further developed to the last detail, in order to allow for third parties, e.g., a machine-tool shop, to produce all the parts leading to the physical prototype upon assembly. Finally, the physical prototype is subjected to validation tests before it is certified and ready to go either into mass production or to the client as an end user, thereby completing the design process. Current trends dictate that the design contemplate not only delivery to the end user, but also disposability of the designed object upon completion of its life cycle.

Engineering design problems have increasingly become model-based, in that their complexity calls for mathematical models involving many quantities, some of which

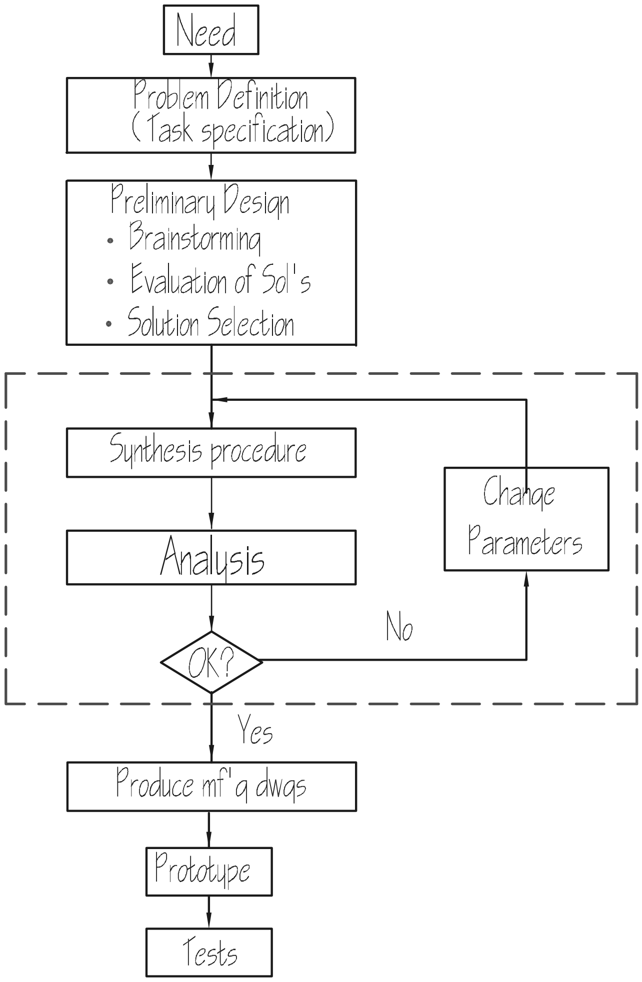


Figure 1.1: The role of optimization within the design process

are to be decided on by the designer with the purpose of meeting performance specifications—e.g., the thrust that an aircraft engine must deliver at a given rpm—under given environment conditions—engine must operate at a specified ambient temperature and at a given ambient pressure. We thus classify the various quantities occurring in the model into:

- *Design Variables (DV):* Those quantities that the designer has to find so as to produce the specified performance under the given conditions;

- *Design-Environment Parameters (DEP):* Those quantities over which the designer has no control, and that define the conditions under which the designed object must operate; and

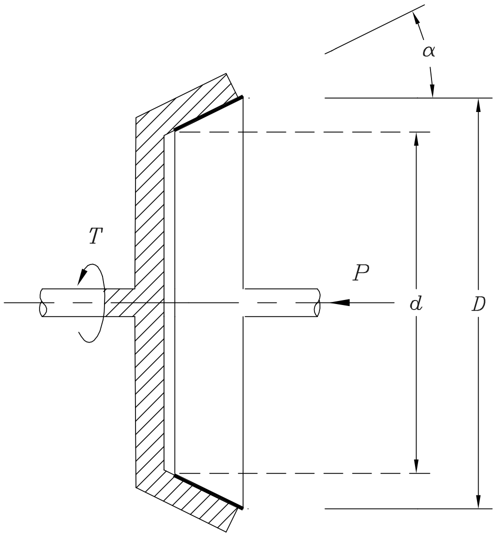


Figure 1.2: A conic clutch

- *Performance Functions (PF):* Relations representing the performance of the design in terms of design variables and design-environment parameters.

Henceforth we shall denote by $\mathbf{x}$ the n-dimensional vector of design variables; we shall refer to $\mathbf{x}$, consequently, as the *design-variable vector*. Likewise, we shall denote by $\mathbf{p}$ the ν-dimensional vector of design-environment parameters, while the μ performance functions, $f_i = f_i(\mathbf{x}; \mathbf{p})$, for $i = 1, 2, \ldots, \mu$ are grouped in the *design-performance vector* $\mathbf{f}$. We thus have

$$\mathbf{x} \equiv \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{p} \equiv \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_\nu \end{bmatrix}, \quad \mathbf{f} \equiv \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_\mu \end{bmatrix}, \quad \begin{aligned} \mathbf{f} &= \mathbf{f}(\mathbf{x}; \mathbf{p}) \\ \mu &< n \end{aligned} \tag{1.1}$$

Needless to say, the DV and the DEP vectors being independent from each other, their dimensions are also independent, and hence, $n \neq \nu$, in general. By the same token, the number μ of performance functions is independent from n and ν. Moreover, DV, DEP and PF will all be termed *design quantities*.

1.1.1 Example 1: The Design of a Conic Clutch

Let us consider the design of the conic clutch shown in Fig. 1.2. The equation involving the design quantities of a clutch is known to be (Shigley and Mischke, 1989)

$$\sin\alpha = \frac{\pi}{8}\frac{\mu P d}{T}(D^2 - d^2) \tag{1.2}$$

where α is the cone angle, μ is the friction coefficient, P is the force applied by the user by means of a pedal, D and d are the outer and inner diameters of the cone, correspondingly, and T is the torque applied on the clutch. Apparently, the designer is responsible for the values of d, D and, to some extent, of μ. However, μ is highly dependent on wear, over which the designer has no control, and hence, it is preferably to consider μ as a DEP; obviously, neither has the designer control on the engine-supplied torque T and the pressure P provided by means external to the clutch itself.

We have thus described a design problem with DV vector $\mathbf{x}$ and design PF f given by

$$\mathbf{x} \equiv \begin{bmatrix} d \\ D \end{bmatrix}^T, \quad f \equiv \sin\alpha = \frac{\pi}{8}\frac{\mu P d}{T}(D^2 - d^2) \tag{1.3}$$

In this example, then, the DEP are the friction coefficient μ, the applied force P and the applied torque T. Hence, the DEP vector is $\mathbf{p} = [\ \mu \quad P \quad T\ ]^T$, and $n = 2,\ \nu = 3,\ \mu = 1$.

Different values of parameters μ, p and T lead to different values of α.

1.1.2 Example 2: The Design of a Low-Pass Filter

We consider now the design of the RL circuit shown in Fig. 1.3, first used by Taguchi (1988) to illustrate the concept of *robust design*, and then by Wilde (1992) to counter Taguchi's approach. The design variables are the resistance R and the inductance L, to be determined by the designer. Moreover, the excitation voltage $v(t)$ is given by $v(t) = V_o \cos\omega t$. The voltage amplitude V_o and its frequency ω undergo variations beyond the control of the designer.

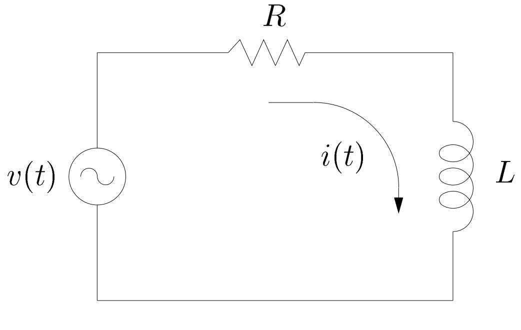


Figure 1.3: A low-pass filter

For this filter, the steady-state current $i(t)$ is harmonic: $i(t) = I_o \cos(\omega t + \phi)$, where I_o and ϕ are the magnitude and the phase of $i(t)$. These are given by

$$I_o = \frac{V_o}{\sqrt{R^2 + \omega^2 L^2}}\,, \qquad \phi = \tan^{-1}\left(\frac{\omega L}{R}\right)$$

Furthermore, V_0 is 110 volt, while ω is 60 Hz, as provided by a power utility, the filter being designed for an amplitude I_0 of the current of 10 ampere. While it is up to the designer to decide which values to choose for R and L, the voltage amplitude V_0 and frequency ω are subject to variations that lie beyond the control of the designer. By the same token, the filter performance is completely described by the current amplitude I_0 and phase ϕ. The vectors of DV, DEP and PF are, thus,

$$\mathbf{x} = \begin{bmatrix} R \\ L \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} V_o \\ \omega \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} I_o \\ \phi \end{bmatrix}$$

Apparently, then, in this design problem we have $n = 2$, $\nu = 2$ and $\mu = 2$.

1.2 The Structure of Optimum Design Problems

In optimum design problems, the DEP are given by the client either explicitly or implicitly. The designer then assumes that these bear *ideal values* that are *representative* of the operation conditions, and do not change. The fact of the matter is that the DEP entail values that are random and hence, are known, if at all, only through their statistics, such as mean values and standard deviations; when the parameters obey a Gaussian distribution, mean value and standard deviation are sufficient to describe the variation of the DEP. In this case, then, the designer uses the mean values of the DEP as the *nominal values* of these parameters, on which the design is based.

The foregoing approach is classical, and will be followed here. An alternative approach, due to Taguchi (1988), consists in admitting that the DEP are not fixed, but varying in a random manner, beyond the control of the designer, the purpose of the design task then being to select the design variables in such a way that, under arbitrary variations of the DEP within a certain range, the PF exhibit "small" variations. This approach is known as *robust design*. For an introductory course, we will not dwell on this approach. Suffice it to say that the methods discussed in this course are equally applicable to robust design, if with a suitable alternative formulation.

In an optimum design problem the designer chooses either one or several PF and formulates an *objective function* to be either minimized, when this function represents a cost, or maximized, when the same represents a profit. As a matter of fact, profit-maximization can be readily turned into cost-minimization if the profit is redefined as a cost by, for example, reversing its sign or taking its reciprocal. Moreover, a large class of optimum design problems lends itself to a *least-square* formulation, which inherently aims at *minimizing* a sum of squares. For these reasons, and unless otherwise stated, we will aim in this course at the *minimization* of an objective function $f(\mathbf{x})$. Defining the objective function is thus a simple matter when only one of the PF is of interest. In defining objective functions, however, as in many design tasks, it will prove convenient to use dimensionless quantities.

1.2.1 Example 3: The Optimum Design of an IC Engine Valve-Driving Mechanism

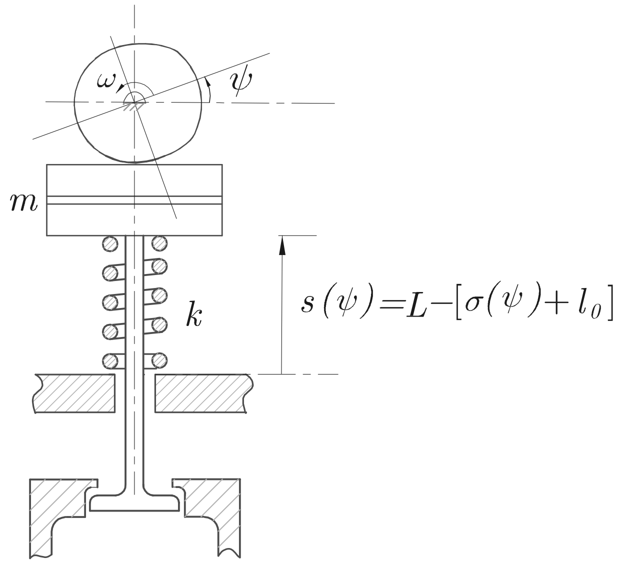


Figure 1.4: Cam mechanism

In this problem we want to determine how "stiff" the valve-opening mechanism of an IC engine, shown in Fig. 1.4, should be designed so as to satisfy the operation conditions described below. In order to simplify the ensuing mathematical model, we neglect friction forces; then, we find the *minimum* natural frequency $\omega_n = \sqrt{k/m}$ of the mechanism, that will prevent separation between cam and follower, for a given cam-speed $\dot{\psi} = \omega_0 = \text{const}$.

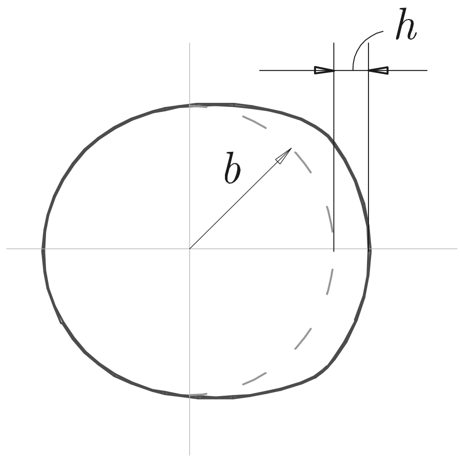


Figure 1.5: Cam profile

We shall refer to the cam geometry of Fig. 1.5, where b is the radius of the *base circle* and h is the *rise*. The cam profile has been found so as to produce the variable part $\sigma(\psi)$ of the follower-displacement program of Fig. 1.6. According to this program, the follower rises from $\psi = 70^\circ = 7\pi/18$ to $\psi = 5\pi/6$; then, dwells till $\psi = 7\pi/6$, to return to its lower dwell at $\psi = 29\pi/18$, which ends after the cam has undergone a rotation of $\psi = 70^\circ = 7\pi/18$. Moreover, both rise and return phases are specified as identical here, while following a cycloidal motion (Angeles and López-Cajún, 1991).

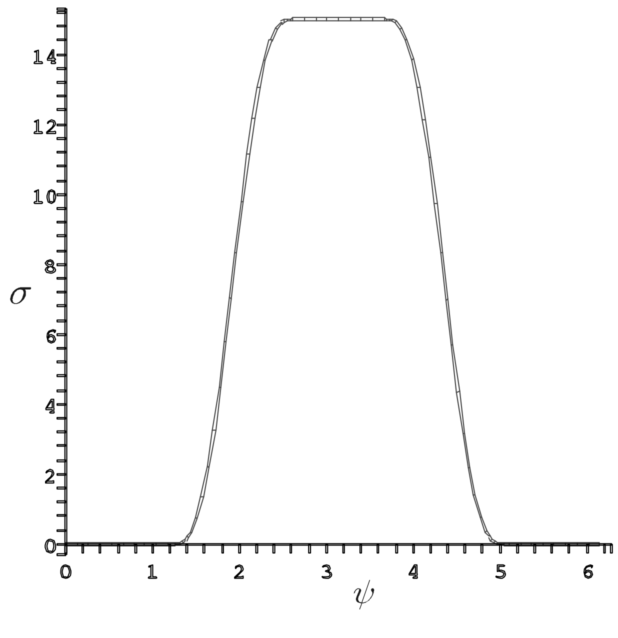


Figure 1.6: The follower-displacement program

For starters, we need a mathematical model, that can be readily derived from a free-body diagram (FBD), as sketched in Fig. 1.7. In this figure, $F(t)$, or $F(\psi)$ for that matter, is the contact force between cam and follower and $k(\sigma + l_0)$ is the force exerted by the spring on the follower, of mass m, under the assumption that the spring is compressed by an amount l_0 when the follower is at the lower dwell. Moreover, we assume that $kl_0 = mg$, with g denoting the gravity acceleration. From the FBD, we have

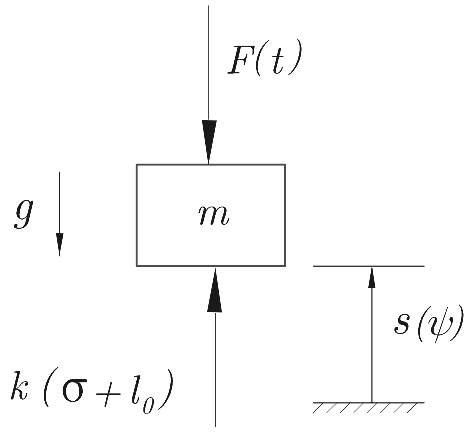


Figure 1.7: Free-body diagram of a cam mechanism

$$m\ddot{s} = -F(t) + k(\sigma + l_0) - mg, \quad s(t) = L - [\sigma(t) + l_0]$$

where L is the length of the undeformed spring. Recalling our assumptions, the foregoing model reduces to

$$F(t) = k\sigma + m\ddot{\sigma}$$

Since the cam turns at a constant speed ω_0, application of the "chain rule" leads to

$$\dot{\sigma} = \sigma'(\psi)\dot{\psi} = -\omega_0\sigma'(\psi)$$

and hence,

$$\ddot{\sigma} = -\omega_0^2\sigma''(\psi)$$

After simple calculations,

$$F(\psi) = m[\omega_n^2\sigma(\psi) + \omega_0^2\sigma''(\psi)]$$

Since ω_0 is a known constant, $F(\psi)$ can be made positive for any value of ψ by choosing ω_n "large enough." Given that $\omega_n^2 = k/m$, ω_n can be rendered high by choosing a "large" value of k and a "small" value of m. However, choosing k too large will require an extremely large volume to accommodate the mechanism. By the same token, a too-small value of m will lead quickly to wear and increase unnecessarily the maintenance costs of the engine. Hence, ω_n should be made "large enough," but not "too large." Just how large is large-enough is the subject of the discussion below.

In order to have $F(\psi) \geq 0$, we must have $\omega_n^2 \sigma(\psi) + \omega_0^2 \sigma''(\psi) \geq 0$, i.e.,

$$\frac{\omega_n^2}{\omega_0^2} \geq -\frac{\sigma''(\psi)}{\sigma(\psi)}$$

Apparently, the minimum value of ω_n guaranteeing that no separation will occur is that which makes the above relation an equality for the *maximum* value of the right-hand side, i.e.,

$$\frac{\omega_n^2}{\omega_0^2} = \max_{\psi} \left\{ -\frac{\sigma''(\psi)}{\sigma(\psi)} \right\}$$

the problem now reducing to finding the maximum of the ratio $-\sigma''(\psi)/\sigma(\psi)$. It is apparent that this ratio takes its maximum in the region where $\sigma''(\psi) \leq 0$, and hence, the search for the maximum can be constrained to the return stage, i.e., to the interval $[7\pi/6,\, 29\pi/18]$. Hence, we can state our optimum-design problem in the form

$$\max_{\psi} \left\{ -\frac{\sigma''(\psi)}{\sigma(\psi)} \right\}$$

subject to

$$7\pi/6 \leq \psi \leq 29\pi/18$$

Alternatively, the foregoing problem can be stated as

$$f(\psi) \equiv -\frac{\sigma''(\psi)}{\sigma(\psi)} \quad \rightarrow \quad \max_{\psi}$$

subject to

$$7\pi/6 \leq \psi \leq 8\pi/6$$

We will not dwell on the methods of solution of the foregoing problem here. The intention is just to highlight a few features of the example at hand:

1. A design problem has led to the maximization of a real function f of one single variable, ψ;

2. The maximization is *constrained* to a finite interval on the real axis;

3. The real function $f(\psi)$ to be maximized *is not analytic* over the whole real axis, which means that $f(\psi)$ does not admit a series expansion *everywhere* on the real axis, and hence, it does not have continuous derivatives everywhere, of an arbitrary order[1].

The above features are common of optimization problems, although,

1. An optimization problem usually involves the minimization or maximization of a real function of many variables, which is called a multivariable-optimization problem; and

2. sometimes the function to minimize or to maximize is analytic everywhere; even if not analytic everywhere, it is analytic within a subinterval in which the minimum or, correspondingly, the maximum, finds itself. In our case, $f(\psi)$ is analytic in $]7\pi/6,\ 8\pi/6[$, which excludes the extremes of the subinterval.

[1]As a matter of fact, $f(\psi)$ itself is continuous and admits continuous first- and second-order derivatives everywhere, but its third- and higher-order derivatives are discontinuous at the blending points of the dwells with the rise and the return stages.

Chapter 2

Single-Variable Optimization

2.1 Methods of Single-Variable Optimization

While real-life design problems involve multiple variables, some techniques developed to find the optimum of these problems rely on a search along each of the variables at a time. Moreover, the designer in many instances is interested in the role played by one single variable, in which case the search for the optimum value can be conducted with techniques specific to this case. For this reason, it is convenient to study techniques applicable to the solution of single-variable optimization problems, which is the subject of this chapter. We start by introducing a **definition:**

> *A function $f(x)$ is* unimodal *in the interval $[\,0,\,1\,]$ if it attains one single extremum—a minimum or a maximum—within this interval.*

Remarks:

- We will deal only with function minimization in explaining the methods of interest—function maximization can be handled by paraphrasing the corresponding method accordingly.
- A unimodal function need neither be continuous nor smooth.
- Defining the interval of interest as $[\,0,\,1\,]$ is not restrictive. If this interval is $[\,a,\,b\,]$, where a and b are any real numbers, then a simple linear transformation of the variable in question can lead to the above interval.

We introduce, moreover, the **basic assumption:** Function $f(x)$, to be minimized, is *unimodal* in the interval $[\,0,\,1\,]$, which means that $f(x)$ attains exactly one minimum (or one maximum) in the given interval.

As a consequence of the above definition, we have

Lemma 2.1.1 *Let $f(x)$ be unimodal in $[\,0,\,1\,]$ and attain a minimum within this interval. Then, its maximum lies necessarily at the* extremes *of the interval, i.e., either at $x = 0$ or at $x = 1$.*

The proof of this lemma is left to the reader as an exercise. Moreover, note that:

- The objective function can be evaluated *only at a discrete, finite set of sample values* of its argument x, $\{\,x_i\,\}_1^n$. Each function evaluation, $f(x_i) \equiv f_i$, is termed an *experiment*. The name is quite appropriate because in some instances it may happen that the evaluation of function $f(x)$ can be done only by physical experiments, e.g., when this function is the steady-state temperature of an engine, that is known to change as the proportion of a mixture of fuel and air varies;

- We assume that the interval in which the minimum lies is *known*, and termed the *interval of uncertainty* (i.o.u) of the problem at hand. Upon a suitable transformation of the design variable, this interval is mapped into the *normal interval* $[\,0,\,1\,]$, which is of unit length. The length of the interval of uncertainty when the series of experiments is initiated is thus 1, the purpose of the minimization exercise being to bring down the interval of uncertainty to an acceptable low, which is dictated mostly by the cost of each experiment;

- If the cost of each experiment is not an issue, then the function can be evaluated in a rich sample of argument values within the interval $[\,0,\,1\,]$ and plot the corresponding values; the optimum can then be located by inspection, possibly at the click of a mouse. This is termed an *exhaustive search*;

- If the foregoing cost is high, then proceed *iteratively*: At each iteration, the interval of uncertainty is cut by a certain factor using a suitable *strategy*, i.e., a search method;

- Any strategy exploits the unimodality assumption. We can cite four strategies that are the most commonly employed:

  - Dichotomous search
  - Interval-halving
  - Fibonacci numbers
  - Golden search.

2.2 Dichotomous Search

The qualifier "dichotomous" derives from Greek, meaning *to cut into two parts.* The strategy to follow thus consists in splitting the interval into two subintervals, not necessarily of the same length, with one not containing the minimum, and is hence, rejected; the other subinterval then is bound to contain the minimum sought.

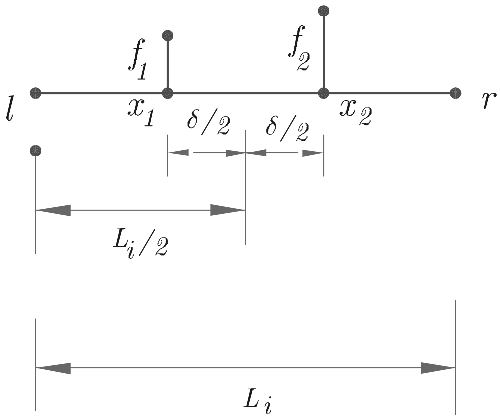


Figure 2.1: Dichotomous search

The search strategy of this method is described below:

- Assume that, at iteration i, the current search interval is $\mathcal{I}_i = [\,l,\, r\,]$, of length $L_i = r - l < 1$ (Fig.2.1);

- locate **two** points of abscissae x_1 and x_2 around the centre of the interval: For a "small" $\delta > 0$, prescribed by the user,

$$x_1 \equiv \frac{r + l - \delta}{2}, \quad x_2 \equiv \frac{r + l + \delta}{2}; \quad f_i \equiv f(x_i); \quad f_1 \neq f_2$$

  Note: If $f_1 = f_2$, then we have two cases: (i) $f(x)$ is symmetric about $x = (r + l)/2$, in which case the minimum lies at $x = 1/2$, and we are done; and (ii) $f(x)$ is not symmetric about $x = 1/2$, in which case we just change δ.

- if $f_2 > f_1$, then eliminate the interval segment to the right of x_2, the new search interval being $[\,l,\, x_2\,]$. If, on the contrary, $f_1 > f_2$, then eliminate the interval segment to the left of x_1, the new search interval being $[\,x_1,\, r\,]$.

Notice that the new search interval $\mathcal{I}_{i+1}$ is of length $L_{i+1} = (L_i + \delta)/2$, i.e., slightly over one half the length of the previous one.

Now we determine the length L_{2k} of interval $\mathcal{I}_{2k}$ after $2k$ experiments—this number is always even! To this end, we notice how the length of the i.o.u. evolves as the search progresses:

$$\begin{aligned} L_2 &= \frac{1}{2} + \frac{\delta}{2} \\ L_4 &= \frac{L_2}{2} + \frac{\delta}{2} = \frac{1}{4} + \frac{\delta}{4} + \frac{\delta}{2} \\ &= \frac{1}{4} + \frac{3\delta}{4} \end{aligned}$$

$$
\begin{aligned}
L_6 &= \frac{L_4}{2} + \frac{\delta}{2} = \frac{1}{8} + \frac{3\delta}{8} + \frac{\delta}{2} \\
&= \frac{1}{8} + \frac{7\delta}{8} \\
&\vdots
\end{aligned}
$$

The length of the interval after $2k$ experiments is thus

$$
L_{2k} = \frac{1}{2^k} + \left(1 - \frac{1}{2^k}\right)\delta \tag{2.1}
$$

Usually, L_{2k} is prescribed, but k is not. Computing k from L_{2k} is, nevertheless, straightforward, as described below: Solving eq.(2.1) for 2^k yields

$$
2^k = \frac{1-\delta}{L_{2k} - \delta} \tag{2.2}
$$

and hence,

$$
k = \left\lceil \frac{\ln[(1-\delta)/(L_{2k}-\delta)]}{\ln(2)} \right\rceil \tag{2.3}
$$

where $\lceil(\,\cdot\,)\rceil$ is the *ceiling function*, defined as the smallest integer that is greater than the real argument $(\,\cdot\,)$. Note: $1 > \delta,\ 2^k > 0\ \Rightarrow\ L_{2k} > \delta$.

2.2.1 Example 4: Finding the Maximum Dexterity Posture of a Two-Phalanx Robotic Finger

We study here the optimum dimensioning of the two-phalanx robotic finger, as depicted in Fig. 2.2. The geometry of the finger is thus completely specified by the angles that the phalanx axes make with given lines. In the figure, only the angle θ made by the distal phalanx with the first one is indicated because only this angle is relevant to the problem under study.

In optimizing the performance of robotic hands, one is interested in maximizing their *dexterity*, a performance index that comes into play as explained below.

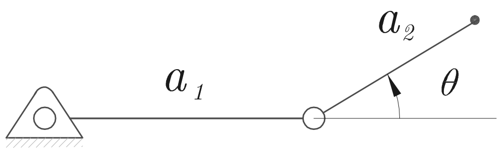


Figure 2.2: Two-phalanx robotic finger

In robot control, a velocity $\mathbf{v}$ of the *operation point* of the end link is to be produced by a suitable set of joint rates, grouped in vector $\dot{\mathbf{q}}$, the relation between the two vectors being linear: $\mathbf{J}\dot{\mathbf{q}} = \mathbf{v}$. Hence, the *Jacobian matrix* $\mathbf{J}(\mathbf{q})$ must be inverted in order to compute the joint-rate vector, for a given posture of the manipulator, as specified by vector $\mathbf{q}$, and a given desired velocity $\mathbf{v}$. Dexterity measures, essentially, how *invertible* the Jacobian matrix is. If we assume that $a_1 = l$ and $a_2 = l\sqrt{2}/2$, which bear the optimum proportion found by Salisbury and Craig (1982), then dexterity can be quantified by means of the product $\mathbf{J}^T\mathbf{J} = \ell^2\mathbf{K}$, where $\mathbf{K}$ is given by

$$\mathbf{K} \equiv \begin{bmatrix} 3 + 2\sqrt{2}\cos\theta & 1 + \sqrt{2}\cos\theta \\ 1 + \sqrt{2}\cos\theta & 1 \end{bmatrix}$$

It should be apparent that, when $\theta = 0$ or π, matrix $\mathbf{K}$, that we shall term here the *dexterity matrix*, is singular, and hence, not invertible, as is $\mathbf{J}$. Between these two values, 0 and π, there is one specific value θ_o optimum, at which the dexterity matrix is *maximally invertible*. To find θ_o, we start by defining the dexterity as the ratio of the smallest (λ_m) to the largest (λ_M) eigenvalues of $\mathbf{K}$. In this regard, note that $\mathbf{K}$ is symmetric, and hence, its eigenvalues are real. Moreover, one can readily verify that $\mathbf{K}$ is positive-definite, and becomes singular only for the two values of θ given above. We thus have the dexterity function $D(\theta)$ defined below:

$$D(\theta) = \frac{\lambda_m}{\lambda_M} \geq 0, \quad 0 \leq D(\theta) \leq 1$$

Now, maximizing $D(\theta)$ is equivalent to minimizing $f(\theta) \equiv 1 - D(\theta)$, which will be defined as the *loss of dexterity*, and becomes, then, the *objective function* of the problem at hand. Given the form of the objective function, then, each experiment involves four steps:

1. For a given value of θ, compute the two eigenvalues of $\mathbf{K}$, a task that can be readily implemented using an eigenvalue routine, a quadratic-equation solver, or even the Mohr circle (Norton, 2000).

2. Order the two eigenvalues in ascending order: λ_m, λ_M.

3. Compute $D(\theta)$ as
$$D(\theta) = \frac{\lambda_m}{\lambda_M} \geq 0$$

4. Compute $f(\theta)$ as
$$f(\theta) = 1 - D(\theta) \quad 0 \leq f(\theta) \leq 1$$

   An interpretation of $f(\theta)$ can be obtained if we redefine the *loss of dexterity* $L(\theta)$ in the form

$$L(\theta) \equiv f(\theta) \times 100\ (\%) = [1 - D(\theta)] \times 100\ (\%), \quad 0 \leq L(\theta) < 100\%$$

and hence, when $D(\theta)$ attains its maximum value of unity, $L(\theta)$ attains its minimum of 0. At the other end of the spectrum, when the finger is postured at a singularity, $L(\theta)$ attains its maximum of 100%, indicating that the finger has lost all its dexterity.

An expert roboticist claims that the dexterity is maximum—the finger is at the peak of its positioning accuracy—when θ lies "somewhere between 90° and 150°." Find an estimate of θ_{opt} within an interval of uncertainty of 5% of the given interval length of 60°.

Solution: We implemented the dichotomous search in the Maple worksheet described below, which is posted in the course Web page.

```
> restart:  with(linalg):
```

We start by producing a procedure K that will allow us to evaluate matrix **K** for a given value θ:

```
> K:=proc(theta)
matrix([[3+2*sqrt(2)*cos(th),1+sqrt(2)*cos(th)],
[1+sqrt(2)*cos(th),  1]])
end;
```

$K :=$ **proc**(θ)
  matrix([[$3 + 2 * \text{sqrt}(2) * \cos(th)$, $1 + \text{sqrt}(2) * \cos(th)$], [$1 + \text{sqrt}(2) * \cos(th)$, 1]])
end proc

```
> argu:= 3*Pi/4; K(argu);
# Testing procedure, which should yield the 2 by 2 identity matrix for
this value of argument theta:
```

$$argu := \frac{3}{4}\pi$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Apparently, procedure is OK.

We introduce now a transformation that maps θ, given in degrees, into the normal interval $[\,0,\,1\,]$. Let the associated "normal" variable be x, to be produced by a second procedure x. By the same token, we need a third procedure θ to return the angle in radians, for a given value of x. Thus,

```
> x:=proc(th)(th-Pi/2)/(5*Pi/6-Pi/2)
end;
```

$$x := \textbf{proc}(th)\, 3*(th - 1/2*\pi)/\pi\, \textbf{end proc}$$

```
> thet:=proc(x)(Pi/3)*x+Pi/2
end;
```

$$thet := \textbf{proc}(x)\, 1/3*\pi*x + 1/2*\pi\, \textbf{end proc}$$

Now we determine the number $2k$ of experiments needed to attain the prescribed length of the i.o.u. We recall that the length L_{2k} of this interval is given by

$$L_{2k} = \frac{1}{2^k} + \delta\left(1 - \frac{1}{2^k}\right) \tag{2.4}$$

```
> L[2*k]:=(1/2^k)+delta*(1-(1/2^k));
```

$$L_{2\,k} := \frac{1}{2^k} + \delta\,(1 - \frac{1}{2^k})$$

Let $2^k = N$. Then,

```
> N:=solve(L_N= (1/N) + delta*(1 - 1/N), N);
```

$$N := \frac{-1+\delta}{-L\_N+\delta}$$

We want the length of the final i.o.u. to be 5% of original length, for a value of δ of 0.01:

```
> delta:=0.01; L_N:= 0.05;
```

$$\delta := 0.01$$

$$L\_N := 0.05$$

```
> N:=subs((delta=0.01, L_N=0.05), N);
```

$$N := 24.75000000$$

Hence,

```
> k:=ceil(solve(2^k=N,k));
```

$$k := 5$$

where the Maple `ceil(·)` command has been used.

We thus need $2k = 10$ experiments. Hence, the two points x_1 and x_2 within I_0 are defined as

```
> x[1]:= (1-delta)/2; x[2]:=(1+delta)/2;
```

$$x_1 := .4950000000$$

$$x_2 := .5050000000$$

```
> theta[1]:=evalf(thet(x[1]));
theta[2]:=evalf(thet(x[2]));
K1:=evalf(K(theta[1]));
K2:=evalf(K(theta[2]));
```

$$\theta_1 := 2.089159115$$

$$\theta_2 := 2.099631090$$

$$K1 := \begin{bmatrix} 1.598631263 & .2993156313 \\ .2993156313 & 1. \end{bmatrix}$$

$$K2 := \begin{bmatrix} 1.572980385 & .2864901922 \\ .2864901922 & 1. \end{bmatrix}$$

```
> lambda:=eigenvals(K1); mu:=eigenvals(K2);
```

$$\lambda := .8760194061, 1.722611857$$

$$\mu := .8813318770, 1.691648508$$

```
> f[1]:=lambda[2]/lambda[1]; f[2]:=mu[2]/mu[1];
```

$$f_1 := 1.966408330$$

$$f_2 := 1.919422810$$

$f_1 > f_2 \Rightarrow$ delete subinterval $[0, x_1]$. Let l and r denote, respectively, the abscissae of the left and right ends of the new subinterval:

```
> l:=x[1]; r:=1; L:=r-l;
```

$$l := .4950000000$$

$$r := 1$$

$$L := .5050000000$$

L is length of i.o.u. at the end of the first two experiments. Carry on:

```
> x[1]:=(l+r-delta)/2; x[2]:=(l+r+delta)/2;
```

$$x_1 := 0.7425000000$$

$$x_2 := 0.7525000000$$

```
> theta[1]:=evalf(thet(x1));
theta[2]:=evalf(thet(x2));
K1:=evalf(K(theta[1])); K2:=evalf(K(theta[2]));
```

$$\theta_1 := 2.348340509$$

$$\theta_2 := 2.358812484$$

$$K1 := \begin{bmatrix} 1.015769486 & .0078847430 \\ .0078847430 & 1. \end{bmatrix}$$

$$K2 := \begin{bmatrix} .994770873 & -.002614564 \\ -.002614564 & 1. \end{bmatrix}$$

```
> lambda:=eigenvals(K1); mu:=eigenvals(K2);
```

$$\lambda := .9967340325,\ 1.019035453$$

$$\mu := .9936878850,\ 1.001082988$$

```
> f[1]:=lambda[2]/lambda[1]; f[2]:=mu[2]/mu[1];
```

$$f_1 := 1.022374495$$

$$f_2 := 1.007442078$$

$f_1 > f_2 \Rightarrow$ delete subinterval $[l,\ x_1]$. Redefine l and r:

```
> l:=x[1]; L:=r-l; #r remains unchanged
```

$$l := .7425000000$$

$$L := .2575000000$$

L is length of i.o.u. at the end of 3rd & 4th experiments. Carry on:

```
> x[1]:=(l+r-delta)/2; x[2]:=(l+r+delta)/2;
```

$$x_1 := .8662500000$$

$$x_2 := .8762500000$$

```
> theta[1]:=evalf(thet(x[1]));
theta[2]:=evalf(thet(x[2]));
K1:=evalf(K(theta[1]));     K2:=evalf(K(theta[2]));
```

$$\theta_1 := 2.477931206$$

$$\theta_2 := 2.488403181$$

$$K1 := \begin{bmatrix} .771929030 & -.114035485 \\ -.114035485 & 1. \end{bmatrix}$$

$$K2 := \begin{bmatrix} .753805937 & -.123097032 \\ -.123097032 & 1. \end{bmatrix}$$

```
> lambda:=eigenvals(K1); mu:=eigenvals(K2);
```

$$\lambda := .7246939855,\ 1.047235044$$

$$\mu := .7028174767,\ 1.050988460$$

```
> f[1]:=lambda[2]/lambda[1]; f[2]:=mu[2]/mu[1];
```

$$f_1 := 1.445072079$$

$$f_2 := 1.495393178$$

The ensuing computations follow the same pattern. In the interest of brevity, we record here only the last two experiments:
At the end of 7th and 8th experiments, we have

$$r := .8143750000$$

$$L := .0718750000$$

```
> x[1]:=(l+r-delta)/2; x[2]:=(l+r+delta)/2;
```

$$x_1 := 0.7734375000$$

$$x_2 := 0.7834375000$$

```
> theta[1]:=evalf(thet(x[1]));
theta[2]:=evalf(thet(x[2]));
K1:=evalf(K(theta[1]));      K2:=evalf(K(theta[2]));
```

$$\theta_1 := 2.380738183$$

$$\theta_2 := 2.391210159$$

$$K1 := \begin{bmatrix} .951519906 & -.024240047 \\ -.024240047 & 1. \end{bmatrix}$$

$$K2 := \begin{bmatrix} .931208945 & -.034395528 \\ -.034395528 & 1. \end{bmatrix}$$

```
> lambda:=eigenvals(K1); mu:=eigenvals(K2);
```

$$\lambda := .9414793498,\ 1.010040556$$

$$\mu := .9169618507,\ 1.014247094$$

```
> f[1]:=lambda[2]/lambda[1]; f[2]:=mu[2]/mu[1];
```

$$f_1 := 1.072822847$$

$$f_2 := 1.106095192$$

$f_2 > f_1 \Rightarrow$ delete subinterval $[\ x_2, r]$. Redefine l and r:

```
> r:=x[2]; L:=r-l; #l remains unchanged
```

$$r := .7834375000$$

$$L := .0409375000$$

L is length of i.o.u. at the end of 9th & 10th experiments. Since L is smaller than 0.05, we're done. The best estimate of θ_{opt} is obviously the mid point of current i.o.u., i.e.,

```
> x[opt]:=(l+r)/2; theta:=evalf(thet(x[opt]));
```

$$x_{opt} := .7629687500$$

$$\theta := 2.369775334$$

```
> K[opt]:=evalf(K(theta));
```

$$K_{opt} := \begin{bmatrix} .973023585 & -.013488208 \\ -.013488208 & 1. \end{bmatrix}$$

```
> lambda:=eigenvals(K[opt]);
```

$$\lambda := .9674365862,\ 1.005586999$$

```
> f[o]:=lambda[2]/lambda[1];
```

$$f_o := 1.039434536$$

```
> theta[opt]:=evalf(th*180/Pi);
```

$$\theta_{opt} := 135.7781250$$

Note that optimum value of theta is 135°, i.e., $3\pi/4$ rad, which yields a value of f=1.0.

2.3 Interval-Halving

To be included.

2.4 Fibonacci Numbers

Fibonacci numbers are named after the Italian mathematician Leonardo Pisano (1175), son of Guglielmo Bonaccio, and hence, referred to as *Filius Bonacci* in Latin, or *Fibonacci* for brevity (Livio, 2002). These numbers form a *sequence*, defined *recursively* as

$$F_0 = F_1 = 1 \tag{2.5a}$$

$$F_k = F_{k-2} + F_{k-1} \tag{2.5b}$$

Remark: The sequence is *monotonically increasing*, for all numbers are positive integers and the current one equals the sum of the two previous ones. From eq.(2.5b),

$$F_k - F_{k-1} = F_{k-2} \tag{2.6}$$

Moreover, by virtue of the above remark,

$$F_{k-1} > F_{k-2} \tag{2.7}$$

Upon addition of eq. (2.6) to inequality (2.7) sidewise, we obtain

$$F_k - F_{k-1} + F_{k-1} > 2F_{k-2}$$

i.e.,

$$\frac{F_{k-2}}{F_k} < \frac{1}{2} \tag{2.8}$$

Furthermore, from eq.(2.5b),

$$\frac{F_{k-1}}{F_k} = 1 - \frac{F_{k-2}}{F_k} \tag{2.9}$$

Now we outline the strategy to follow in this method:

- Let $\mathcal{I}_0 \equiv [\,0,\,1\,]$ be the initial interval of uncertainty, of length 1, where the minimum is *known* to lie.
- **Prescribe** the number n of experiments to be conducted
- **Define** a length L_2^* as[1]

$$L_2^* \equiv \frac{F_{n-2}}{F_n} L_0 \equiv \frac{F_{n-2}}{F_n} \quad \left(< \frac{1}{2}\right) \tag{2.10}$$

where L_0 is the length of the original interval, which has been defined as unity.

[1]Length L_1^* is skipped because we want to make the subscript of L^* match that of the corresponding Fibonacci number; since $F_1 = F_0$, the first two Fibonacci numbers are undistinguishable, and we can arbitrary set $L_1^* = L_2^* = 1$. $L_1 = L_0$ as well.

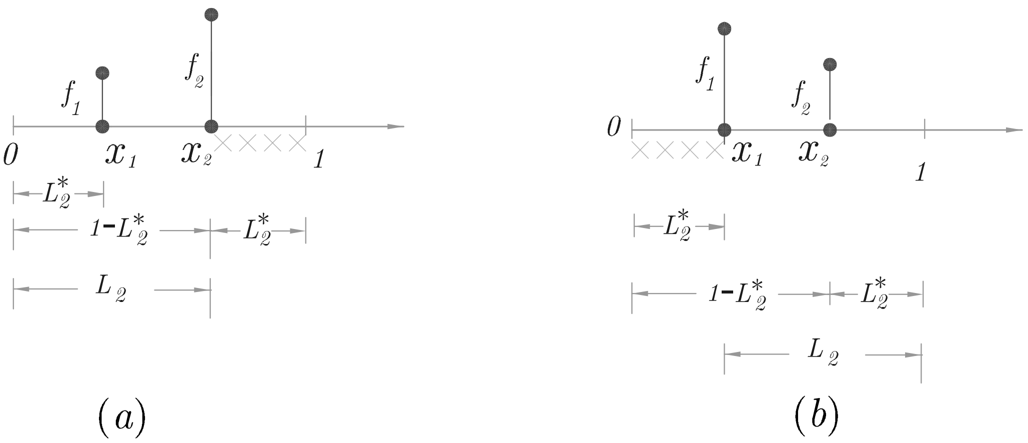


Figure 2.3: Shrinking of the i.o.u. when: (a) right subinterval is eliminated; (b) left subinterval is eliminated

Figure 2.4: Subdivision of the new interval into three subintervals when: (a) right subinterval is eliminated; (b) left subinterval is eliminated

- **Let** $P_1(x_1)$ and $P_2(x_2)$ be two points *equidistant* from the left and the right ends of $\mathcal{I}_0$, respectively, by a distance L_2^*, i.e., with abscissae

$$x_1 = 0 + L_2^* = \frac{F_{n-2}}{F_n} < \frac{1}{2} \tag{2.11a}$$

$$x_2 = 1 - L_2^* = 1 - \frac{F_{n-2}}{F_n} > \frac{1}{2} > x_1 \tag{2.11b}$$

  Note that

$$x_2 = 1 - \frac{F_{n-2}}{F_n} = \frac{F_n - F_{n-2}}{F_n} = \frac{F_{n-1}}{F_n} \tag{2.11c}$$

Use the unimodality assumption to eliminate the subinterval, left or right, where the minimum **cannot** lie. Whether the subinterval eliminated is the right or the left, the length L_2 of the new, shorter interval $\mathcal{I}_2$—again, such as we do not define

L_1^*, we neither define $\mathcal{I}_1$—is given by

$$L_2 = 1 - L_2^* = 1 - \frac{F_{n-2}}{F_n} = \frac{F_n - F_{n-2}}{F_n} = \frac{F_{n-1}}{F_n} < 1 \tag{2.12}$$

as depicted in Fig.2.3.

Once the first iteration is completed, and as illustrated in Fig.2.3,

- Let l and $r > l$ denote, respectively, the abscissae of the left and the right ends of the current, smaller interval of uncertainty $\mathcal{I}_2$. The abscissa of one of the ends of $\mathcal{I}_2$, left or right, is either x_1 or x_2;

- if x_1 is the abscissa of one of the ends of $\mathcal{I}_2$, case of Fig. 2.3(b), then $x_2 \in \mathcal{I}_2$; else, as in Fig. 2.3(a), $x_1 \in \mathcal{I}_2$. Let x_I be the abscissa of point P_I, the interior point of $\mathcal{I}_2$, either P_1 or P_2;

- note that P_I lies a distance L_3^* from one of the ends of $\mathcal{I}_2$. Now, define $x_3 \in \mathcal{I}_2$, so that its associated point P_3 also lies a distance L_3^* from the other end, as depicted in Fig. 2.4;

- the process is continued until the interval $\mathcal{I}_n$, of length L_n, is obtained. L_n is the length of the final interval of uncertainty.

The abscissa x_j computed at the jth experiment is determined by length L_j^*, so that its associated point P_j, as well as the interior point P_I of interval $\mathcal{I}_{j-1}$ are a distance L_j^* from the ends of $\mathcal{I}_j$, with L_j^* given by[2]

$$L_j^* = \frac{F_{n-j}}{F_{n-(j-2)}} L_{j-1} \tag{2.13a}$$

while the length L_j of the jth interval of uncertainty is[3]

$$L_j = \frac{F_{n-(j-1)}}{F_n} \tag{2.13b}$$

Hence, for $j = n$,

$$L_n = \frac{F_1}{F_n} \tag{2.14}$$

which allows us to find n for a prescribed length L_n. Notice that, by virtue of relations (2.8) and (2.10),

$$\text{For} \quad n \geq 2, \quad L_n^* < \frac{1}{2} \tag{2.15}$$

[2]see eq.(2.10), which is valid for $j = 2$; see also footnote 1.

[3]see eq.(2.12), which is valid for $j = 2$

The Location of the Final Experiment

Let $\mathcal{I}_{n-1} = [\,l,\, r\,]$, of length L_{n-1}, be the one-before-the-last interval of uncertainty. According with eqs.(2.11a & b), the abscissae of the last two experiments, x_{n-1} and x_n, are given as

$$x_{n-1} = l + L_n^* \tag{2.16a}$$

$$x_n = r - L_n^* = l + L_{n-1} - L_n^* \tag{2.16b}$$

where L_n^* is given by eq.(2.13a), with $j = n$:

$$L_n^* = \frac{F_0}{F_2} L_{n-1} \equiv \frac{1}{2} L_{n-1} \tag{2.17}$$

Upon substitution of L_n^*, as given by eq.(2.17), into eq.(2.16b), it is apparent that

$$x_n = l + \frac{1}{2} L_{n-1} = x_{n-1}$$

x_{n-1} and x_n thus coinciding, and hence, the last experiment fails to produce two distinct points in $\mathcal{I}_{n-1}$. To cope with this outcome, we have to define points P_1 and P_2, of abscissae x_1 and x_2, in an alternative manner. For example, we can define them as in the strategy employed by the dichotomous search, with a δ small enough with respect to L_n^*.

Fibonacci numbers are tabulated in many manuals, with short tables available in textbooks (Rao, 1996). Also note that scientific software is provided with Fibonacci numbers. For example, Maple includes the command

```
with(combinat, Fibonacci):
```

that allows the user to invoke the Fibonacci number `F(i)` by typing

```
fibonacci(i)
```

However, note that **not all Fibonacci sequences are identical**. For example, the first two Fibonacci numbers in Maple are defined as

`f(0)` $=$ `0` and `f(1)` $=$ `1`

2.4.1 Example 5: Finding the Maximum Dexterity Posture of a Two-Phalanx Robotic Finger

We implement the Fibonacci search in a Maple worksheet:

```
> restart:
> with(linalg):  with(combinat, fibonacci):
```

We retake the example problem consisting in the finding of the most dexterous posture of a two-phalanx robotic finger, using exactly **10 experiments.** However, because of the way Maple defines the Fibonacci sequence, we must use $F(n+1)$ when we would normally use $F(n)$. Moreover, we shall also use K, θ and x exactly as described in Subsection 2.2.1.

We want to have

$$L_2^* = \frac{F_{10-2}}{F_{10}}, \tag{2.18}$$

but must shift the subscript by 1:

```
> Lstar[2]:=evalf(fibonacci(9)/fibonacci(11));
```

$$Lstar_2 := .3820224719$$

```
> l:=0; r:=1; L[0]:=r - l;
> # Abscissae of extremes of left- & right-hand
> sides of the initial (normal) interval, and length of this interval
```

$$l := 0$$

$$r := 1$$

$$L_0 := 1$$

```
> x[1]:= l + L^star[2]; x[2]:= r - Lstar[2];
```

$$x_1 := .3820224719$$

$$x_2 := .6179775281$$

```
> theta[1]:=evalf(thet(x1));
> theta[2]:=evalf(thet(x2));
> K1:=K(theta[1]);   K2:=K(theta[2]);
```

$$\theta_1 := 1.970849324$$

$$\theta_2 := 2.217940881$$

$$K1 := \begin{bmatrix} 3 - .7789343108\sqrt{2} & 1 - .3894671554\sqrt{2} \\ 1 - .3894671554\sqrt{2} & 1 \end{bmatrix}$$

$$K2 := \begin{bmatrix} 3 - 1.205821535\sqrt{2} & 1 - .6029107675\sqrt{2} \\ 1 - .6029107675\sqrt{2} & 1 \end{bmatrix}$$

```
> lambda:=<eigenvals(K1)>;
> mu:=<eigenvals(K2)>;
```

$$\lambda := \begin{bmatrix} .8139310151 \\ 2.084489519 \end{bmatrix}$$

$$\mu := \begin{bmatrix} .9389633883 \\ 1.355747444 \end{bmatrix}$$

```
> f[1]:=lambda[2]/lambda[1];
> f[2]:=mu[2]/mu[1];
```

$$f_1 := 2.561014976$$

$$f_2 := 1.443876791$$

$f_1 > f_2 \Rightarrow$ drop left end:

```
> l:=x[1]; L[2]:=L[0] - Lstar[2];
> Lstar[3]:=(fibonacci(8)/fibonacci(10))*L[2];
```

$$l := .3820224719$$

$$L_2 := .6179775281$$

$$Lstar_3 := .2359550562$$

```
> x[1]:= l + Lstar[3]; x[2]:= r - Lstar[3];
```

$$x_1 := .6179775281$$

$$x_2 := .7640449438$$

```
> theta[1]:=evalf(thet(x[1]));
> K1:=K(theta[1]);
> theta[2]:=evalf(thet(x[2])); K2:=K(theta[2]);
```

$$\theta_1 := 2.217940881$$

$$K1 := \begin{bmatrix} 3 - 1.205821535\sqrt{2} & 1 - .6029107675\sqrt{2} \\ 1 - .6029107675\sqrt{2} & 1 \end{bmatrix}$$

$$\theta_2 := 2.370902321$$

$$K2 := \begin{bmatrix} 3 - 1.434859867\sqrt{2} & 1 - .7174299337\sqrt{2} \\ 1 - .7174299337\sqrt{2} & 1 \end{bmatrix}$$

```
> lambda:=<eigenvals(K1)>;
> mu:=<eigenvals(K2)>;
```

$$\lambda := \begin{bmatrix} .9389633883 \\ 1.355747444 \end{bmatrix}$$

$$\mu := \begin{bmatrix} .9647545542 \\ 1.006047163 \end{bmatrix}$$

```
> f[1]:=lambda[2]/lambda[1];
> f[2]:=mu[2]/mu[1];
```

$$f_1 := 1.443876791$$

$$f_2 := 1.042801155$$

$f_1 > f_2 \Rightarrow$ drop left end:

```
> l:= x[1]; L[3]:= L[2] - Lstar[3];
> Lstar[4]:= (fibonacci(7)/fibonacci(9))*L[3];
```

$$l := .6179775281$$

$$L_3 := .3820224719$$

$$Lstar_4 := .1460674157$$

```
> x[1]:= l + Lstar[4]; x[2]:= r - Lstar[4];
```

$$x_1 := .7640449438$$

$$x_2 := .8539325843$$

```
> theta[1]:=evalf(thet(x[1])); K1:=K(theta[1]);
> theta[2]:=evalf(thet(x[2])); K2:=K(theta[2]);
```

$$\theta[1] := 2.370902321$$

$$K1 := \begin{bmatrix} 3 - 1.434859867\sqrt{2} & 1 - .7174299337\sqrt{2} \\ 1 - .7174299337\sqrt{2} & 1 \end{bmatrix}$$

$$\theta_2 := 2.465032438$$

$$K2 := \begin{bmatrix} 3 - 1.559462054\sqrt{2} & 1 - .7797310268\sqrt{2} \\ 1 - .7797310268\sqrt{2} & 1 \end{bmatrix}$$

```
> lambda:=<eigenvals(K1)>;
> mu:=<eigenvals(K2)>;
```

$$\lambda := \begin{bmatrix} .9647545542 \\ 1.006047163 \end{bmatrix}$$

$$\mu := \begin{bmatrix} .7520453159 \\ 1.042542298 \end{bmatrix}$$

```
> f[1]:=lambda[2]/lambda[1];
> f[2]:=mu[2]/mu[1];
```

$$f_1 := 1.042801155$$

$$f_2 := 1.386275901$$

In the interest of brevity, we skip the intermediate results, and display only the last two experiments:

```
> x[1]:= l + Lstar[8]; x[2]:= r - Lstar[8];
```

$$x_1 := .7303370787$$

$$x_2 := .7415730337$$

```
> theta[1]:=evalf(thet(x1)); K1:=K(theta[1]);
> theta[2]:=evalf(thet(x2)); K2:=K(theta[2]);
```

$$\theta_1 := 2.335603527$$

$$K1 := \begin{bmatrix} 3 - 1.384795807\sqrt{2} & 1 - .6923979033\sqrt{2} \\ 1 - .6923979033\sqrt{2} & 1 \end{bmatrix}$$

$$\theta_2 := 2.347369792$$

$$K2 := \begin{bmatrix} 3 - 1.401678651\sqrt{2} & 1 - .7008393253\sqrt{2} \\ 1 - .7008393253\sqrt{2} & 1 \end{bmatrix}$$

```
> lambda:=<eigenvals(K1)>;
> mu:=<eigenvals(K2)>;
```

$$\lambda := \begin{bmatrix} .9913837386 \\ 1.050219250 \end{bmatrix}$$

$$\mu := \begin{bmatrix} .9963286091 \\ 1.021398433 \end{bmatrix}$$

```
> f[1]:=lambda[2]/lambda[1]; f[2]:=mu[2]/mu[1];
```

$$f_1 := 1.059346859$$

$$f_2 := 1.025162204$$

```
> l:= x[1]; L[8]:= L[7] - Lstar[8];
> Lstar[9]:= (fibonacci(2)/fibonacci(4))*L[8];
```

$$l := .7303370787$$

$$L_8 := .03370786516$$

$$Lstar_9 := .01123595505$$

Note that the length of the i.o.u. at the end of the 8th experiment is 3.4% of original length, i.e., smaller than at the end of 10 experiments with the dichotomous search!

```
> x[1]:= l + Lstar[9]; x[2]:= r - Lstar[9];
```

$$x_1 := .7415730338$$

$$x_2 := .7528089888$$

```
> theta[1]:=evalf(thet(x[1])); K1:=K(theta[1]);
> theta[2]:=evalf(thet(x[2]));
> K2:=K(theta[2]);
```

$$\theta_1 := 2.347369792$$

$$K1 := \begin{bmatrix} 3 - 1.401678651\sqrt{2} & 1 - .7008393253\sqrt{2} \\ 1 - .7008393253\sqrt{2} & 1 \end{bmatrix}$$

$$\theta_2 := 2.359136057$$

$$K2 := \begin{bmatrix} 3 - 1.418367442\sqrt{2} & 1 - .7091837208\sqrt{2} \\ 1 - .7091837208\sqrt{2} & 1 \end{bmatrix}$$

```
> lambda:=<eigenvals(K1)>;
> mu:=<eigenvals(K2)>;
```

$$\lambda := \begin{bmatrix} .9963286091 \\ 1.021398433 \end{bmatrix}$$

$$\mu := \begin{bmatrix} .9929088850 \\ 1.001216643 \end{bmatrix}$$

```
> f[1]:=lambda[2]/lambda[1];
> f[2]:=mu[2]/mu[1];
```

$$f_1 := 1.025162204$$

$$f_2 := 1.008367090$$

$f_1 > f_2 \Rightarrow$ drop left end:

```
> l:= x[1]; L[9]:= L[8] - Lstar[9];
> Lstar[10]:= (fibonacci(1)/fibonacci(3))*L[9];
```

$$l := .7415730338$$

$$L_9 := .02247191011$$

$$Lstar_{10} := .01123595506$$

```
> x[1]:= l + Lstar[10]; x[2]:= r - Lstar[10];
```

$$x_1 := .7528089889$$

$$x_2 := .7528089887$$

As expected, $x_1 = x_2$. Let us estimate the optimum by dichotomous search over the last i.o.u.: Let $\delta = Lstar_{10}/10$

```
> delta:= Lstar[10]/10;
```

$$\delta := .001123595506$$

```
> x[1]:= (l + r - delta)/2;
> x[2]:= (l + r +  delta)/2;
```

$$x_1 := .7522471910$$

$$x_2 := .7533707866$$

```
> theta[1]:=evalf(thet(x[1])); K1:=K(theta[1]);
> theta[2]:=evalf(thet(x[2])); K2:=K(theta[2]);
```

$$\theta_1 := 2.358547744$$

$$K1 := \begin{bmatrix} 3 - 1.417537647\sqrt{2} & 1 - .7087688235\sqrt{2} \\ 1 - .7087688235\sqrt{2} & 1 \end{bmatrix}$$

$$\theta_2 := 2.359724370$$

$$K2 := \begin{bmatrix} 3 - 1.419196745\sqrt{2} & 1 - .7095983727\sqrt{2} \\ 1 - .7095983727\sqrt{2} & 1 \end{bmatrix}$$

```
> lambda:=<eigenvals(K1)>;
> mu:=<eigenvals(K2)>;
```

$$\lambda := \begin{bmatrix} .9943254329 \\ 1.000973602 \end{bmatrix}$$

$$\mu := \begin{bmatrix} .9914931757 \\ 1.001459540 \end{bmatrix}$$

```
> f[1]:=lambda[2]/lambda[1];
> f[2]:=mu[2]/mu[1];
```

$$f_1 := 1.006686110$$

$$f_2 := 1.010051874$$

$f_2 > f_1 \Rightarrow$ delete $[\,x_2,\, r\,]$ and take as most likely estimate of the optimum the midpoint of remaining interval $[\,l,\, x_1\,]$:

```
> x[o]:=(l+x[1])/2; th_opt:=evalf(th(x[o]));
> K_opt:= K(th_opt);
```

$$x_o := .7469101124$$

$$\theta_{opt} := 2.352958768 \longrightarrow 134.8146°$$

$$Kopt := \begin{bmatrix} 3 - 1.409630165\sqrt{2} & 1 - .7048150824\sqrt{2} \\ 1 - .7048150824\sqrt{2} & 1 \end{bmatrix}$$

```
> lambda:=<eigenvals(Kopt)>;
```

$$\lambda := \begin{bmatrix} .9986575537 \\ 1.007824349 \end{bmatrix}$$

```
> f[opt]:=lambda[2]/lambda[1];
```

$$f_{opt} := 1.009179118$$

2.5 Golden-Section Search

This method is similar to the method based on Fibonacci numbers, but its implementation is much simpler. The outcome is that its convergence is a bit slower than that of the former. A major difference with the Fibonacci search is that the number of elimination stages is not prescribed.

The basis of the golden-search method is the Fibonacci sequence. Indeed, the golden-search strategy is derived from the Fibonacci search under the assumption that n in the Fibonacci search is "large"; we denote a large n appropriately by N. The length of the interval of uncertainty is shrunken at every iteration by the same proportion, as opposed to the Fibonacci search.

In order to find the length L_k of the interval $\mathcal{I}_k$ at the kth iteration of the golden search, we compute the corresponding lengths of the Fibonacci search for $n = N \to \infty$, namely,

$$L_2 = \lim_{N\to\infty} \frac{F_{N-1}}{F_N} \tag{2.19a}$$

$$L_3 = \lim_{N\to\infty} \frac{F_{N-2}}{F_N} = \lim_{N\to\infty} \frac{F_{N-2}}{F_{N-1}}\frac{F_{N-1}}{F_N} = \lim_{N\to\infty} \left(\frac{F_{N-1}}{F_N}\right)^2 \tag{2.19b}$$

In general,

$$L_k = \lim_{N\to\infty} \left(\frac{F_{N-1}}{F_N}\right)^{k-1} \tag{2.19c}$$

Hence, all we need to implement this method is the above limit, which is computed below:

Recall eq.(2.5b), for $k = N$:

$$F_N = F_{N-1} + F_{N-2} \tag{2.20a}$$

Therefore,

$$\frac{F_N}{F_{N-1}} = 1 + \frac{F_{N-2}}{F_{N-1}} \tag{2.20b}$$

Now we define the *golden section* or *golden ratio* ϕ as

$$\phi \equiv \lim_{N\to\infty} \frac{F_N}{F_{N-1}} \tag{2.20c}$$

Upon taking limits, eq.(2.20b) can be rewritten as

$$\phi = 1 + \frac{1}{\phi}$$

or

$$\phi^2 - \phi - 1 = 0, \quad \phi > 0 \tag{2.20d}$$

whence, with three decimals,

$$\phi = 1.618 \quad \text{or} \quad \phi = -0.618 \tag{2.20e}$$

Obviously, we need only the positive root, and hence, L_k becomes

$$L_k = \left(\frac{1}{\phi}\right)^{k-1} = (0.618)^{k-1} \tag{2.21}$$

Greeks in the classical period, around the fifth century B.C.E., coined the expression *golden section* to refer to a rectangle of *divine proportions*, whose base b and height h observe the relation

$$\frac{b+h}{b} = \frac{b}{h} \tag{2.22}$$

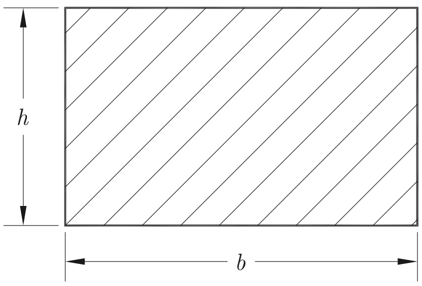


Figure 2.5: A rectangle with sides obeying the divine proportion

The foregoing equation readily leads to one on the ratio b/h identical to eq.(2.20d), namely,

$$\left(\frac{b}{h}\right)^2 - \frac{b}{h} - 1 = 0$$

thereby showing that the positive solution to eq.(2.22) is, indeed, ϕ, the golden section.

This relation is claimed to appear in the facade of the Parthenon, although Livio (2002) disputed brilliantly this claim and others along the same lines. Nevertheless, the golden section is irrefutably present in nature and in many artifacts[4]. Shown in Fig. 2.5 is a rectangle with sides obeying the divine proportion.

To implement the search, we need the quantity L_2^*, which is defined below:

$$L_2^* = \frac{F_{N-2}}{F_N} = \frac{F_{N-2}}{F_{N-1}}\frac{F_{N-1}}{F_N} = \frac{1}{\phi^2} = 0.382 \tag{2.23}$$

[4]For example, Microsoft Word uses the golden-section ratio to proportion its margins.

2.5.1 Example 6: Finding the Maximum Dexterity Posture of a Two-Phalanx Robotic Finger

We implement below the golden-section search strategy by means of a Maple worksheet. We shall resort to the K, θ and x procedures introduced earlier.

Let us calculate ϕ:

```
> eq:=(x^2 - x - 1);
```

$$eq := x^2 - x - 1$$

```
> r:=<solve(eq, x)>;
```

$$r := \begin{bmatrix} \frac{1}{2} + \frac{1}{2}\sqrt{5} \\ \frac{1}{2} - \frac{1}{2}\sqrt{5} \end{bmatrix}$$

```
> phi:=evalf(r[1]);
```

$$\phi := 1.618033989$$

```
> ihp :=1.0/phi; #we'll also need the reciprocal of phi
```

$$ihp := .6180339887$$

```
> Lstar[2]:=ihp ^2;
```

$$Lstar_2 := .3819660112$$

Now let us find n from the problem specification: $L_n = 0.05$, which leads to

$$\frac{1}{\phi^{n-1}} = 0.05 \tag{2.24}$$

```
> eq:= (n-1)*ln(ihp ) - ln(0.05)=0;
```

$$eq := -.4812118251\,n + 3.476944099 = 0$$

```
> n:=ceil(solve(eq, n));
```

$$n := 8$$

```
> l:=0; r:=1; #extremes of initial normal
> interval
```

$$l := 0$$

$$r := 1$$

```
> L[0]:= r - l; #length of initial interval
```

$$L_0 := 1$$

```
> x[1]:=l+Lstar[2]; x[2]:=r-Lstar[2];
```

$$x_1 := .3819660112$$

$$x_2 := .6180339888$$

```
> theta[1]:=evalf(thet(x[1])); K1:=K(theta[1]);
> theta[2]:=evalf(thet(x2)); K2:=K(theta[2]);
> lambda:=<eigenvals(K1)>; mu:=<eigenvals(K2)>;
```

$$\theta_1 := 1.970790199$$

$$K1 := \begin{bmatrix} 3 - .7788253964\sqrt{2} & 1 - .3894126982\sqrt{2} \\ 1 - .3894126982\sqrt{2} & 1 \end{bmatrix}$$

$$\theta_2 := 2.218000007$$

$$K2 := \begin{bmatrix} 3 - 1.205915875\sqrt{2} & 1 - .6029579377\sqrt{2} \\ 1 - .6029579377\sqrt{2} & 1 \end{bmatrix}$$

$$\lambda := \begin{bmatrix} .8138991148 \\ 2.084675447 \end{bmatrix}$$

$$\mu := \begin{bmatrix} .9389910200 \\ 1.355586395 \end{bmatrix}$$

```
> f[1]:=lambda[2]/lambda[1];
> f[2]:=mu[2]/mu[1];
```

$$f_1 := 2.561343794$$

$$f_2 := 1.443662789$$

$f_1 > f_2 \Rightarrow$ drop the left end:

```
> L[2]:=L[0] - Lstar[2]; Lstar[3]:= x[2] - x[1];
> # You should be able to prove that ihp ^3 = x_2 - x_1
```

$$L_2 := .6180339888$$

$$Lstar_3 := .2360679776$$

```
> x3:= r - Lstar[3];
> # x[2] is now a distance  Lstar_3 from new left end, x[1],
> to the left of  x[3]
```

$$x_3 := .7639320224$$

```
> theta[3]:=evalf(th(x[3])); K3:=K(theta[3]);
> lambda:=<eigenvals(K3)>;
```

$$\theta_3 := 2.370784070$$

$$K3 := \begin{bmatrix} 3 - 1.434695103\sqrt{2} & 1 - .7173475515\sqrt{2} \\ 1 - .7173475515\sqrt{2} & 1 \end{bmatrix}$$

$$\lambda := \begin{bmatrix} .9650358237 \\ 1.005998904 \end{bmatrix}$$

```
> f[3]:=lambda[2]/lambda[1];f[2];
> #evaluate f[3]  & recall f[2]
```

$$f_3 := 1.042447212$$

$$f_2 := 1.443662789$$

$f_2 > f_3 \Rightarrow$ drop left end:

```
> L[3]:=L[2] - Lstar[3];
> Lstar[4]:= x3 - x2;
```

$$L_3 := .3819660112$$

$$Lstar_4 := .1458980336$$

```
> x[4]:= r - Lstar[4];
```

$$x_4 := .8541019664$$

```
> theta[4]:=evalf(thet(x[4])); K4:=K(th4);
> lambda:=<eigenvals(K4)>;
```

$$\theta_4 := 2.465209815$$

$$K4 := \begin{bmatrix} 3 - 1.559684146\sqrt{2} & 1 - .7798420728\sqrt{2} \\ 1 - .7798420728\sqrt{2} & 1 \end{bmatrix}$$

$$\lambda := \begin{bmatrix} .7516661806 \\ 1.042607347 \end{bmatrix}$$

```
> f[4]:=lambda[2]/lambda[1];f[3];
> #evaluate f[4]  & recall f[3]
```

$$f_4 := 1.387061669$$

$$f_3 := 1.042447212$$

Again, for brevity we introduce only the last two experiments. We have the interval $[\,x_5,\,x_6\,]$ and hence,

```
> L[6]:=L[5] - Lstar[6]; Lstar[7]:= x6 - x3;
```

$$L_6 := .0901699440$$

$$Lstar_7 := .0344418544$$

```
> x[7]:= x[5] + Lstar[7];
```

$$x_7 := .7426457872$$

```
> theta[7]:=evalf(thet(x[7]));
> K7:=K(theta[7]);lambda:=<eigenvals(K7)>;
```

$$\theta_7 := 2.348493177$$

$$K7 := \begin{bmatrix} 3 - 1.403280430\sqrt{2} & 1 - .7016402150\sqrt{2} \\ 1 - .7016402150\sqrt{2} & 1 \end{bmatrix}$$

$$\lambda := \begin{bmatrix} .9967977596 \\ 1.018664025 \end{bmatrix}$$

```
> f[7]:=lambda[2]/lambda[1];f[3];
> #evaluate f[7] & recall f[3]:
```

$$f_7 := 1.021936511$$

$$f_3 := 1.042447212$$

$f_7 < f_3 \Rightarrow$ drop right end and perform last experiment:

```
> L[7]:=L[6] - Lstar[7]; Lstar[8]:=x3-x7;
```

$$L_7 := .0557280896$$

$$Lstar_8 := .0212862352$$

```
> x[8]:= x[5] + Lstar[8];
```

$$x_8 := .7294901680$$

```
> theta[8]:=evalf(thet(x8)); K8:=K(theta[8]);
> lambda:=<eigenvals(K8)>;
```

$$\theta_8 := 2.334716645$$

$$K8 := \begin{bmatrix} 3 - 1.383515463\sqrt{2} & 1 - .6917577317\sqrt{2} \\ 1 - .6917577317\sqrt{2} & 1 \end{bmatrix}$$

$$\lambda := \begin{bmatrix} .9910087350 \\ 1.052404934 \end{bmatrix}$$

```
> f[8]:=lambda[2]/lambda[1];f[7];
> #evaluate f[8]  & recall f[7]:
```

$$f_8 := 1.061953237$$

$$f_7 := 1.021936511$$

$f_8 > f_7 \Rightarrow$ delete left end and accept midpoint, of abscissa x_o, as best estimate of optimum:

```
> x[o]:= (x[8] + x[3])/2;
```

$$x_o := .7467110952$$

Evaluate $f_o = f(x_o)$:

```
> theta[o]:=evalf(thet(x[o]));
> Ko:=K(th[o]); lambda:=<eigenvals(Ko)>;
```

$$\theta_o := 2.352750357$$

$$Ko := \begin{bmatrix} 3 - 1.409334444\sqrt{2} & 1 - .7046672222\sqrt{2} \\ 1 - .7046672222\sqrt{2} & 1 \end{bmatrix}$$

$$\lambda := \begin{bmatrix} .9985709394 \\ 1.008329177 \end{bmatrix}$$

```
> L[8]:= x[3] - x[8];
> #Length of final i.o.u.
```

$$L_8 := .0344418544$$

Notice that length of final i.o.u. is 3.4% the length of original i.o.u.

```
> f[opt]:=lambda[2]/lambda[1];
```

$$f_{opt} := 1.009772203$$

```
> theta[opt]:= evalf(thet[o]*180/Pi);
> # theta_optimum in  degrees
```

$$\theta_{opt} := 134.8026657$$

Chapter 3

Numerical Equation Solving

3.1 Introduction

Multivariable optimization frequently calls for the solution of systems of equations that can be linear, nonlinear, or a combination thereof. If linear, then a solution can be found numerically by means of a *direct method*[1], as opposed to *iterative methods*. This is a major difference, because direct methods involve a fixed number of operations; on the contrary, iterative methods involve a fixed number of operations *per iteration*, but the number of iterations the method will take until convergence is reached cannot be predicted. Furthermore, if the nonlinear equations of a system are *algebraic*, i.e., *multivariate polynomials*, then the system can be reduced, at least in principle, to a single univariate polynomial, if of a degree higher than that of any of the individual equations.

When the objective function and the constraints are multivariate polynomials in the design variables, the optimization problem leads to a system of multivariate polynomials, if with extra variables, namely, the Lagrange multipliers, to be introduced in Ch. 5. Under these conditions, it is possible to use elimination methods, as implemented in computer-algebra code, to eliminate all but one of the design variables, thereby ending up with a single univariate polynomial. Having reduced the optimization problem to polynomial-root finding is advantageous, because the roots of the polynomial provide all *stationary points*—as defined in Ch. 2 and Ch. 4, these are points where the objective function ceases to change locally—and, hence,

[1]Some linear systems of a large number of unknowns and weakly coupled equations, frequently arising in some contexts, like structural mechanics, can be solved to a great advantage using an iterative method like Gauss-Seidel's.

all *local minima*. The *global minimum* can then be found by inspection.

As an alternative to univariate-polynomial reduction, the optimization problem at hand can be reduced to two (or more) bivariate equations, polynomial or trigonometric, whose plots appear as contours in the plane of those two variables. All solutions can then be obtained visually, by contour-intersection.

Prior to the discussion of equation-solving, we revisit the fundamental concepts of linear algebra that will be needed in the sequel. Then, we recall the basic problem of solving a system of n linear equations in n unknowns, what is called a *determined system*. The issue of roundoff-error amplification is given due attention, which takes us to the concept of *condition number*.

As a natural extension of the above problem, we undertake the problem of *linear least squares*. That is, we now study the solution of a system of q linear equations in n unknowns, when $q > n$, what is called an *overdetermined* system of linear equations. In this case, in general, it is not possible to find a single vector $\mathbf{x}$ that verifies the *redundant* and, most likely *inconsistent*, set of equations. Hence, we aim at finding the *best fit* in the least-square sense, i.e., the vector $\mathbf{x}$ that approximates the whole set of q equations with the minimum *Euclidean norm*. We derive a closed-form expression, i.e., a *formula*, for the best fit $\mathbf{x}$ directly from the *normality conditions* (NC) of the problem at hand. This derivation readily leads to the *left Moore-Penrose generalized inverse* (LMPGI) of the coefficient matrix, which is rectangular, and for which an *inverse* proper cannot be defined. It is shown that computing the best fit *directly* from the NC is prone to *ill-conditioning*, a phenomenon characterized by a "large" roundoff-error amplification. Hence, the reader is strongly advised against computing the best fit with the said formula. Instead, *orthogonalization algorithms* are to be used. The difference between a formula, like that giving the best fit in terms of the LMPGI, and an *algorithm* is stressed here: The LMPGI is seldom needed as such, in the same way that the inverse of a nonsingular (square) matrix is seldom needed. Therefore, the computation of such a generalized inverse is to be avoided.

3.2 Background Facts and Definitions

We begin by recalling the concept of vector and matrix norms:

A *norm* is to an array of numbers, be it a column vector, a row vector, or a matrix, what the absolute value is to real numbers and the module is to complex numbers.

Vector norms can be defined in various ways:

The *Euclidean norm*: The best known. For a n-dimensional vector $\mathbf{a}$ with components a_i, for $i = 1, \ldots, n$:

$$\|\mathbf{a}\|_E \equiv \sqrt{a_1^2 + \cdots + a_n^2} \tag{3.1}$$

Computing this norm thus requires n multiplications, n additions, and one square root. Not very "cheap" to compute!

The *Chebyshev norm*, a.k.a. the *maximum norm*, or the *infinity norm*:

$$\|\mathbf{a}\|_\infty \equiv \max_i \{|a_i|\}_1^n \tag{3.2}$$

Notice that this norm requires no floating-point operations (flops): quite economical.

The *p-norm*:

$$\|\mathbf{a}\|_p \equiv \left(\sum_{j=1}^{n} |a_i|^p \right)^{1/p} \tag{3.3}$$

This is the most general case. For $p = 2$, the p-norm becomes the Euclidean norm; for $p \to \infty$, the p-norm becomes the Chebyshev norm.

Likewise, matrix norms can be defined in various ways:

- The *Euclidean norm*, a.k.a. the 2-norm: the square root of the largest (non-negative) eigenvalue of the positive-semidefinite product of the matrix by its transpose, regardless of the ordering of the factors. For example, for the $n \times n$ matrix $\mathbf{A}$,

  $$\|\mathbf{A}\|_E \equiv \max_i \{\sqrt{\lambda_i}\} \tag{3.4}$$

  where $\{\ \lambda_i\ \}_1^n$ is the set of non-negative eigenvalues of $\mathbf{A}\mathbf{A}^T$, or of $\mathbf{A}^T\mathbf{A}$ for that matter. This norm is also called the *spectral norm*. Notice that λ_i is identical to the square of the module of the ith eigenvalue of $\mathbf{A}$ itself.

- The *Frobenius norm*: the square root of the sum of the squares of the entries of the matrix. For the same matrix $\mathbf{A}$,

  $$\|\mathbf{A}\|_F \equiv \sqrt{\sum_{j=1}^{n} \sum_{i=1}^{n} a_{ij}^2} \equiv \sqrt{\mathrm{tr}(\mathbf{A}\mathbf{A}^T)} \tag{3.5}$$

- The *Chebyshev* norm or *infinity norm*: the maximum absolute value of the entries of the matrix. For the above matrix $\mathbf{A}$,

$$\|\mathbf{A}\|_\infty \equiv \max_{i,j}\{|a_{ij}|\} \tag{3.6}$$

- The *p-norm*:

$$\|\mathbf{A}\|_p \equiv \left(\sum_{j=1}^{n}\sum_{i=1}^{n}|a_{ij}|^p\right)^{1/p} \tag{3.7}$$

  For $p = 2$, the p-norm becomes the Frobenius norm; for $p \to \infty$, the p-norm becomes, such as in the vector case, the Chebyshev norm.

Remarks:

- The *trace* of $\mathbf{A}$, $\mathrm{tr}(\mathbf{A})$, is defined as the sum of its diagonal entries: $\mathrm{tr}(\mathbf{A}) \equiv \sum_{i=1}^{n} a_{ii}$.
- The counterpart of the vector Euclidean norm is **not** the Euclidean matrix norm, but rather the *Frobenius norm.*
- The counterpart of the vector Chebyshev norm is the *matrix Chebyshev norm.*

Furthermore,

Definition 3.2.1 A $n\times n$ matrix $\mathbf{A}$ is *symmetric* if it equals its transpose: $\mathbf{A} = \mathbf{A}^T$

Definition 3.2.2 A $n \times n$ matrix $\mathbf{A}$ is *skew-symmetric* if it equals *the negative* of its transpose: $\mathbf{A} = -\mathbf{A}^T$

Fact 3.2.1 (The Matrix Cartesian Decomposition) Every $n\times n$ matrix $\mathbf{A}$ can be decomposed into the sum of a symmetric and a skew-symmetric components:

$$\mathbf{A} = \mathbf{A}_s + \mathbf{A}_{ss} \tag{3.8a}$$

$$\mathbf{A}_s = \frac{1}{2}(\mathbf{A} + \mathbf{A}^T) \tag{3.8b}$$

$$\mathbf{A}_{ss} = \frac{1}{2}(\mathbf{A} - \mathbf{A}^T) \tag{3.8c}$$

Equation (3.8a) is termed the *Cartesian decomposition* of $\mathbf{A}$, because of its resemblance with the Cartesian representation of a complex number $\underline{Z}$ as $x + jy$, with $x, y \in \mathbb{R}$and $j = \sqrt{-1}$. Notice that the eigenvalues of the symmetric component $\mathbf{A}_s$ are all real, but those of $\mathbf{A}_{ss}$ are imaginary. Also notice that the Cartesian decomposition is *unique.*

Definition 3.2.3 A quadratic form q of a n-dimensional vector $\mathbf{x}$ is associated with a $n \times n$ matrix $\mathbf{A}$:

$$q \equiv \mathbf{x}^T \mathbf{A} \mathbf{x} \tag{3.9}$$

Fact 3.2.2 The quadratic form associated with a skew-symmetric matrix vanishes identically. That is, if $\mathbf{A} = -\mathbf{A}^T$, then, for any n-dimensional vector $\mathbf{x}$,

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = 0 \tag{3.10}$$

Proof: Note that, since $q \equiv \mathbf{x}^T \mathbf{A} \mathbf{x}$ is a scalar, $q = q^T$, and hence,

$$(\mathbf{x}^T \mathbf{A} \mathbf{x})^T = \mathbf{x}^T \mathbf{A} \mathbf{x}$$

Expanding the left-hand side,

$$\mathbf{x}^T \mathbf{A}^T \mathbf{x} = \mathbf{x}^T \mathbf{A} \mathbf{x}$$

However, by assumption, $\mathbf{A}^T = -\mathbf{A}$, and hence,

$$-\mathbf{x}^T \mathbf{A} \mathbf{x} = \mathbf{x}^T \mathbf{A} \mathbf{x}$$

whence the proof follows immediately.

Definition 3.2.4 A $n \times n$ matrix $\mathbf{A}$ is *positive-definite* (*positive-semidefinite*) if it is symmetric and, for every n-dimensional vector $\mathbf{x}$, the quadratic form $\mathbf{x}^T \mathbf{A} \mathbf{x}$ is greater than (or equal to) zero.

Characterization of positive-definiteness (semidefiniteness): A $n \times n$ (symmetric) matrix $\mathbf{A}$ is *positive-definite* (*positive-semidefinite*) if and only if its eigenvalues are all positive (nonnegative).

Remarks:

- Negative-definiteness and negative-semidefiniteness are defined and characterized likewise;

- If a matrix is neither positive- nor negative-definite, or semidefinite, then it is said to be *sign-indefinite*.

3.3 Background on Linear Transformations

The general form of a *linear transformation* mapping a vector space $\mathcal{U}$ of dimension n into a m-dimensional vector space $\mathcal{V}$ is

$$\mathbf{v} = \mathbf{L}\mathbf{u} \tag{3.11}$$

where $\mathbf{u}$ and $\mathbf{v}$ are n- and m-dimensional vectors, respectively, with $\mathbf{u} \in \mathcal{U}$ and $\mathbf{v} \in \mathcal{V}$. Apparently, $\mathbf{L}$ is a $m \times n$ matrix.

We distinguish two vector subspaces associated with $\mathbf{L}$, namely,

> The *range* of $\mathbf{L}$, denoted by $\mathcal{R}(\mathbf{L})$: the set of vectors $\mathbf{v}$ that are *images* of $\mathbf{u}$ under transformation (3.11). Notice that, if the n columns of $\mathbf{L}$ are not linearly independent, then $\mathcal{R}(\mathbf{L})$ is not all of $\mathcal{V}$, but only a *proper* subspace of it, of dimension $m' < n$, i.e., $\mathcal{R}(\mathbf{L}) \subset \mathcal{V}$. The dimension of $\mathcal{R}(\mathbf{L})$, known as the rank of $\mathbf{L}$, is denoted by $\rho(\mathbf{L})$.
>
> The *nullspace* or *kernel* of $\mathbf{L}$, denoted by $\mathcal{N}(\mathbf{L})$: the set of all vectors $\mathbf{u}$ of $\mathcal{U}$ that are mapped by $\mathbf{L}$ into $\mathbf{0}_m$, the zero of $\mathcal{V}$. The dimension of $\mathcal{N}$ is termed the *nullity* of $\mathbf{L}$, and is denoted by $\nu(\mathbf{L})$. Obviously, $\nu < n$, with $\nu = n$ occurring only when $\mathbf{L} = \mathbf{O}_{mn}$, $\mathbf{O}_{mn}$ denoting the $m \times n$ zero matrix.

A fundamental result of linear algebra follows:

$$\rho(\mathbf{L}) + \nu(\mathbf{L}) = n \tag{3.12}$$

The most frequent linear transformations used in optimum design are studied in the balance of this section. They all pertain to square matrices.

3.3.1 Rotations

A rotation $\mathbf{Q}$ is an *orthogonal* transformation of $\mathcal{U}$ into itself, with a constraint on its determinant, as we shall outline presently. Orthogonality requires that the inverse of $\mathbf{Q}$ be its transpose, i.e.,

$$\mathbf{Q}\mathbf{Q}^T = \mathbf{Q}^T\mathbf{Q} = \mathbf{1} \tag{3.13}$$

where $\mathbf{1}$ denotes the $n \times n$ identity matrix. Hence, taking the determinant of both sides of the above equation,

$$\det(\mathbf{Q}\mathbf{Q}^T) = \det(\mathbf{Q}^T\mathbf{Q}) = \det(\mathbf{Q})^T\det(\mathbf{Q}) = [\det(\mathbf{Q})]^2 = 1$$

whence

$$\det(\mathbf{Q}) = \pm 1$$

A *proper orthogonal matrix* $\mathbf{Q}$ is one whose determinant is positive, and hence,

$$\det(\mathbf{Q}) = +1 \tag{3.14}$$

Proper orthogonal transformations of $\mathcal{U}$ into itself represent rotations about the origin of $\mathcal{U}$.

The best-known rotations are those in two and three dimensions. Thus, for two dimensions, the 2×2 matrix $\mathbf{Q}$ rotating vectors through an angle ϕ ccw takes the form

$$\mathbf{Q} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \tag{3.15a}$$

which can be expressed alternatively as

$$\mathbf{Q} = (\cos\phi)\mathbf{1}_2 + (\sin\phi)\mathbf{E}_2 \tag{3.15b}$$

with $\mathbf{1}_2$ defined as the 2×2 identity matrix and $\mathbf{E}_2$ as a skew-symmetric matrix, namely,

$$\mathbf{E}_2 \equiv \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{3.15c}$$

In three dimensions, the rotation matrix takes the form

$$\mathbf{Q} = \mathbf{e}\mathbf{e}^T + \cos\phi(\mathbf{1}_3 - \mathbf{e}\mathbf{e}^T) + \sin\phi\mathbf{E}_3 \tag{3.16a}$$

where $\mathbf{e}$ is the unit vector indicating the direction of the axis about which the rotation takes place, $\mathbf{1}_3$ is the 3×3 identity matrix, and $\mathbf{E}_3$ is the *cross-product matrix* (CPM) of vector $\mathbf{e}$, expressed as

$$\mathbf{E}_3 \equiv \mathrm{CPM}(\mathbf{e}) \tag{3.16b}$$

The Cross-Product Matrix

We will start by defining the partial derivative of a vector with respect to another vector. This is a matrix, as described below: In general, let $\mathbf{u}$ and $\mathbf{v}$ be vectors of spaces $\mathcal{U}$ and $\mathcal{V}$, of dimensions m and n, respectively. Furthermore, let t be a real variable and f be real-valued function of t, $\mathbf{u} = \mathbf{u}(t)$ and $\mathbf{v} = \mathbf{v}(\mathbf{u}(t))$ being m- and n-dimensional vector functions of t as well, with $f = f(\mathbf{u}, \mathbf{v})$. The derivative of $\mathbf{u}$ with respect to t, denoted by $\dot{\mathbf{u}}(t)$, is an m-dimensional vector whose ith component is the derivative of the ith component of $\mathbf{u}$ in a given basis, u_i, with respect to t. A similar definition follows for $\dot{\mathbf{v}}(t)$. The partial derivative of f with respect to $\mathbf{u}$ is an m-dimensional vector whose ith component is the partial derivative of f with respect to u_i, with a corresponding definition for the partial derivative of f with respect to $\mathbf{v}$. The foregoing derivatives, as all other vectors, will be assumed, henceforth, to be *column* arrays. Thus,

$$\frac{\partial f}{\partial \mathbf{u}} \equiv \begin{bmatrix} \partial f/\partial u_1 \\ \partial f/\partial u_2 \\ \vdots \\ \partial f/\partial u_m \end{bmatrix}, \quad \frac{\partial f}{\partial \mathbf{v}} \equiv \begin{bmatrix} \partial f/\partial v_1 \\ \partial f/\partial v_2 \\ \vdots \\ \partial f/\partial v_n \end{bmatrix} \tag{3.17}$$

Furthermore, the partial derivative of $\mathbf{v}$ with respect to $\mathbf{u}$ is an $n \times m$ array whose (i, j) entry is defined as $\partial v_i/\partial u_j$, i.e.,

$$\frac{\partial \mathbf{v}}{\partial \mathbf{u}} \equiv \begin{bmatrix} \partial v_1/\partial u_1 & \partial v_1/\partial u_2 & \cdots & \partial v_1/\partial u_m \\ \partial v_2/\partial u_1 & \partial v_2/\partial u_2 & \cdots & \partial v_2/\partial u_m \\ \vdots & \vdots & \ddots & \vdots \\ \partial v_n/\partial u_1 & \partial v_n/\partial u_2 & \cdots & \partial v_n/\partial u_m \end{bmatrix} \tag{3.18}$$

Hence, the total derivative of f with respect to $\mathbf{u}$ can be written as

$$\frac{df}{d\mathbf{u}} = \frac{\partial f}{\partial \mathbf{u}} + \left(\frac{\partial \mathbf{v}}{\partial \mathbf{u}}\right)^T \frac{\partial f}{\partial \mathbf{v}} \tag{3.19}$$

If, moreover, f is an explicit function of t, i.e., if $f = f(\mathbf{u}, \mathbf{v}, t)$ and $\mathbf{v} = \mathbf{v}(\mathbf{u}, t)$, then, one can write the total derivative of f with respect to t as

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \left(\frac{\partial f}{\partial \mathbf{u}}\right)^T \frac{d\mathbf{u}}{dt} + \left(\frac{\partial f}{\partial \mathbf{v}}\right)^T \frac{\partial \mathbf{v}}{\partial t} + \left(\frac{\partial f}{\partial \mathbf{v}}\right)^T \frac{\partial \mathbf{v}}{\partial \mathbf{u}} \frac{d\mathbf{u}}{dt} \tag{3.20}$$

The total derivative of $\mathbf{v}$ with respect to t can be written, likewise, as

$$\frac{d\mathbf{v}}{dt} = \frac{\partial \mathbf{v}}{\partial t} + \frac{\partial \mathbf{v}}{\partial \mathbf{u}} \frac{d\mathbf{u}}{dt} \tag{3.21}$$

Example 3.3.1 *Let the components of* $\mathbf{v}$ *and* $\mathbf{x}$ *in a certain reference frame* $\mathcal{F}$ *be given as*

$$[\,\mathbf{v}\,]_{\mathcal{F}} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}, \quad [\,\mathbf{x}\,]_{\mathcal{F}} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{3.22a}$$

Then

$$[\,\mathbf{v} \times \mathbf{x}\,]_{\mathcal{F}} = \begin{bmatrix} v_2 x_3 - v_3 x_2 \\ v_3 x_1 - v_1 x_3 \\ v_1 x_2 - v_2 x_1 \end{bmatrix} \tag{3.22b}$$

Hence,

$$\left[\frac{\partial(\mathbf{v}\times\mathbf{x})}{\partial\mathbf{x}}\right]_{\mathcal{F}} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \tag{3.22c}$$

Henceforth, the partial derivative of the cross product of any 3-dimensional vectors $\mathbf{v}$ and $\mathbf{x}$ will be denoted by the 3×3 matrix $\mathbf{V}$. For obvious reasons, $\mathbf{V}$ is termed the *cross-product matrix* of vector $\mathbf{v}$. Thus, the foregoing cross product admits the alternative representations

$$\mathbf{v}\times\mathbf{x} = \mathbf{V}\mathbf{x} \tag{3.23}$$

Note that given *any* 3-dimensional vector $\mathbf{a}$, its cross-product matrix $\mathbf{A}$ is *uniquely* defined. Moreover, this matrix is skew-symmetric. The converse also holds, i.e., given any 3×3 skew-symmetric matrix $\mathbf{A}$, its associated *vector* is uniquely defined as well. This result is made apparent from Example 3.3.1.

The cross-product matrix is defined only for three-dimensional vectors. Let $\mathbf{a}$ and $\mathbf{v}$ be two arbitrary three-dimensional vectors. We define

$$\mathrm{CPM}(\mathbf{a}) \equiv \frac{\partial(\mathbf{a}\times\mathbf{v})}{\partial\mathbf{v}} \equiv \mathbf{A} \quad \forall\mathbf{v} \tag{3.24}$$

Because of the relation $\mathbf{a}\times\mathbf{v} = -\mathbf{v}\times\mathbf{a}$, the CPM is skew-symmetric.

Rotations in higher dimensions can be defined as well, but then, the axis and the angle of rotation are not unique.

3.3.2 Reflections

Reflections are improper orthogonal matrices, preserving the distance between any two points of the n-dimensional space. For any $n\times n$ reflection $\mathbf{R}$, we have

$$\det(\mathbf{R}) = -1 \tag{3.25}$$

In two dimensions, a reflection $\mathbf{R}$ about a line passing through the origin normal to the unit vector $\mathbf{e}$ maps a vector $\mathbf{p}$ into $\mathbf{p}'$ in the form

$$\mathbf{p}' = \mathbf{p} - 2(\mathbf{p}^T\mathbf{e})\mathbf{e} = (\mathbf{1} - 2\mathbf{e}\mathbf{e}^T)\mathbf{p} \equiv \mathbf{R}\mathbf{p}$$

and hence, the reflection $\mathbf{R}$ sought is given by

$$\mathbf{R} = \mathbf{1} - 2\mathbf{e}\mathbf{e}^T \tag{3.26}$$

In three dimensions, the reflection about a plane passing through the origin, of unit normal $\mathbf{e}$, takes exactly the same form as $\mathbf{R}$ in the two-dimensional case, eq.(3.26). However, in this case, $\mathbf{e}$ is three-dimensional, while $\mathbf{R}$ is of 3×3, and $\mathbf{1}$ is the 3×3 identity matrix.

In all foregoing instances, the reflections are represented by symmetric matrices, and are hence termed *pure reflections*. However, this need not always be the case, for reflections can combine with rotations, thereby yielding a new reflection—notice that the product of a rotation by a pure reflection is a reflection!—but this time, the matrix representing the reflection is no longer symmetric. A rotation can be distinguished from a reflection by the sign of its determinant.

3.3.3 Projections

Henceforth, a *projection* $\mathbf{P}$ means an *orthogonal projection* onto a plane in n dimensions, which we call the *projection plane*. When $n = 2$, the "projection plane" becomes a line in the plane.

Let us consider a plane Π in a n-dimensional space, of unit normal $\mathbf{n}$. Any point P in this space is given by its n-dimensional position vector $\mathbf{p}$. Let the projection of P onto Π be P', which is given by its position vector $\mathbf{p}'$, namely,

$$\mathbf{p}' = \mathbf{p} - (\mathbf{n}^T\mathbf{p})\mathbf{n} = (\mathbf{1} - \mathbf{n}\mathbf{n}^T)\mathbf{p} \equiv \mathbf{P}\mathbf{p} \tag{3.27}$$

where $\mathbf{P}$ is obviously defined as

$$\mathbf{P} \equiv \mathbf{1} - \mathbf{n}\mathbf{n}^T \tag{3.28}$$

Matrix $\mathbf{P}$ is also called a *projector*. A projector $\mathbf{P}$ is represented by a symmetric, singular, *idempotent* matrix. Symmetry is obvious; singularity is less so, but rather straightforward. To prove that $\mathbf{P}$ is singular, all we have to do is prove that its nullspace is non-empty. However, this is so because all vectors $\mathbf{r}$ of the form $\alpha\mathbf{n}$, for a scalar $\alpha \neq 0$, are mapped by $\mathbf{P}$ onto the zero vector. Indeed,

$$\mathbf{P}\mathbf{r} = \alpha\mathbf{P}\mathbf{n} = \alpha(\mathbf{1} - \mathbf{n}\mathbf{n}^T)\mathbf{n} = \alpha(\mathbf{n} - \mathbf{n}) = \mathbf{0}$$

A matrix is idempotent of degree k when it equals its kth power, but is different from any lower power. When $k = 2$, the degree is self-understood and need not be spelled out. To prove idempotency, let us calculate

$$\mathbf{P}^2 = (\mathbf{1} - \mathbf{n}\mathbf{n}^T)(\mathbf{1} - \mathbf{n}\mathbf{n}^T) = \mathbf{1} - 2\mathbf{n}\mathbf{n}^T + \mathbf{n}\underbrace{\mathbf{n}^T\mathbf{n}}_{=1}\mathbf{n}^T = \mathbf{1} - \mathbf{n}\mathbf{n}^T \equiv \mathbf{P}$$

thereby completing the proof.

The foregoing projection has a nullity of 1, its nullspace being spanned by vector $\mathbf{n}$. In three-dimensional space, we can have projections onto a subspace of dimension 1, namely, a line $\mathcal{L}$ passing through the origin and parallel to the unit vector $\mathbf{e}$. In this case, the projection P' of P onto $\mathcal{L}$ is given by

$$\mathbf{p}' = (\mathbf{p}^T\mathbf{e})\mathbf{e} \equiv \mathbf{e}(\mathbf{e}^T\mathbf{p}) = (\mathbf{e}\mathbf{e}^T)\mathbf{p}$$

whence the projection $\mathbf{P}$ sought takes the form:

$$\mathbf{P} = \mathbf{e}\mathbf{e}^T \tag{3.29}$$

Notice that this projection is symmetric, singular and idempotent as well, its nullspace being of dimension two. Indeed, we can find two mutually-orthogonal unit vectors $\mathbf{f}$ and $\mathbf{g}$, lying in a plane normal to $\mathbf{e}$, which are mapped by $\mathbf{P}$ onto the zero vector. These two linearly-independent vectors lie in the nullspace of $\mathbf{P}$. For n dimensions, the projection "plane" can in fact be a subspace of dimension $\nu \leq n - 1$.

Also notice that the projection of eq.(3.27) maps vectors in three-dimensional space onto the nullspace of the *rank-one matrix* $\mathbf{n}\mathbf{n}^T$, while that of eq.(3.29) does so onto the range of the rank-one matrix $\mathbf{e}\mathbf{e}^T$. Now, the range of this matrix is the nullspace of a matrix $\mathbf{A}$ defined as

$$\mathbf{A} \equiv \begin{bmatrix} \mathbf{f}^T \\ \mathbf{g}^T \end{bmatrix} \tag{3.30}$$

where $\mathbf{f}$ and $\mathbf{g}$ are mutually orthogonal unit vectors normal to $\mathbf{e}$. Then, we can define a projector $\mathbf{P}$ in the form

$$\mathbf{P} = \mathbf{1} - \mathbf{A}^T\mathbf{A} = \mathbf{1} - (\mathbf{f}\mathbf{f}^T + \mathbf{g}\mathbf{g}^T) \tag{3.31}$$

This projector maps three-dimensional vectors onto the nullspace of $\mathbf{A}$, which is vector $\mathbf{e}$, as the reader can readily verify.

In general, if we have a full-rank $m \times n$ matrix $\mathbf{A}$, with $m < n$, then, $\operatorname{rank}(\mathbf{A}) = \min\{\, m,\, n\,\} = m$. This means that the m n-dimensional rows of $\mathbf{A}$ are linearly independent. By virtue of the basic relation (3.12), then, $\nu = n - m$. A projector that maps n-dimensional vectors onto the nullspace of $\mathbf{A}$ is defined below:

$$\mathbf{P} = \mathbf{1} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} \tag{3.32}$$

Note that, by virtue of the definition of $\mathbf{f}$ and $\mathbf{g}$, matrix $\mathbf{A}$ of eq.(3.30) produces $\mathbf{A}\mathbf{A}^T = \mathbf{1}_2$, the 2×2 identity matrix.

Exercise 3.3.1 *Prove that* $\mathbf{P}$*, as given by eq.(3.32), is a projector; then prove that its projection maps* n*-dimensional vectors onto the nullspace of* $\mathbf{A}$.

Example 3.3.2 *Let*

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix} \equiv \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \end{bmatrix}$$

The nullspace of $\mathbf{A}$ *is spanned by a unit vector* $\mathbf{u}$ *that can be found as*

$$\mathbf{u} \equiv \frac{\mathbf{b}}{\|\mathbf{b}\|}, \quad \mathbf{b} \equiv \mathbf{a}_1 \times \mathbf{a}_2$$

The projector $\mathbf{P}$ *mapping vectors in three-dimensional space onto the nullspace of* $\mathbf{A}$*, spanned by* $\mathbf{u}$*, is given by*

$$\mathbf{P} = \mathbf{1} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} = \frac{1}{3}\begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & 1 \\ -1 & 1 & 1 \end{bmatrix}$$

In this case,

$$\mathbf{b} = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{u} = \frac{\sqrt{3}}{3}\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$$

Notice that the image of any vector $\mathbf{p} = [\,x,\ y,\ z\,]^T$ *under* $\mathbf{P}$ *can be expressed as the product of a scalar times* $\mathbf{u}$*:*

$$\mathbf{P}\mathbf{p} = \frac{1}{3}\begin{bmatrix} x - y - z \\ -x + y + z \\ -x + y + z \end{bmatrix} = \frac{1}{3}(-x + y + z)\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} = \frac{\sqrt{3}}{3}(-x + y + z)\mathbf{u}$$

3.4 The Numerical Solution of Determined Linear Systems of Equations

We consider the system

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{3.33}$$

where

$\mathbf{A}$: $n \times n$ matrix of *known* coefficients,

$\mathbf{b}$: n-dimensional right-hand side *known* vector,

$\mathbf{x}$: n-dimensional vector of *unknowns*.

Definition 3.4.1 If

$$\det(\mathbf{A}) = 0 \tag{3.34}$$

then $\mathbf{A}$ is said to be **singular**. Otherwise, $\mathbf{A}$ is **nonsingular**.

Fact 3.4.1 If $\mathbf{A}$ is nonsingular, then eq.(3.33) has a *unique solution*, which is given by

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \tag{3.35}$$

Caveat: Never compute $\mathbf{A}^{-1}$ explicitly ... , unless instructed to do so. The inverse is seldom needed as such, and incurs a waste of precious CPU time! Instead, find a *good* numerical approximation to the solution, while taking into account that $\mathbf{A}$ and $\mathbf{b}$ are usually known only up to a random roundoff error.

Avoid roundoff-error amplification!

3.4.1 Roundoff Error of the Solution and Condition Numbers

Regarding the roundoff-error amplification when solving the system (3.33), let $\delta\mathbf{A}$ be the matrix roundoff error in $\mathbf{A}$, $\delta\mathbf{b}$ be the vector roundoff-error in $\mathbf{b}$, and $\delta\mathbf{x}$ be the vector roundoff-error incurred when solving eq.(3.33), by virtue of $\delta\mathbf{A}$ and $\delta\mathbf{b}$.

The *relative roundoff errors* in the data, $\epsilon_{\mathbf{A}}$ and $\epsilon_{\mathbf{b}}$, and in the computed solution, $\epsilon_{\mathbf{x}}$, are defined as

$$\epsilon_{\mathbf{A}} \equiv \frac{||\delta\mathbf{A}||}{||\mathbf{A}||}, \quad \epsilon_{\mathbf{b}} \equiv \frac{||\delta\mathbf{b}||}{||\mathbf{b}||}, \quad \epsilon_{\mathbf{x}} \equiv \frac{||\delta\mathbf{x}||}{||\mathbf{x}||} \tag{3.36}$$

where $||\cdot||$ denotes *any* vector or matrix norm.

The relative roundoff error in the computed solution is known to be related to the relative roundoff error in the data via the relation (Golub and Van Loan, 1983)

$$\epsilon_{\mathbf{x}} \leq \kappa(\mathbf{A})(\epsilon_{\mathbf{A}} + \epsilon_{\mathbf{b}}) \tag{3.37}$$

with $\kappa(\mathbf{A})$ defined as the *condition number* of matrix $\mathbf{A}$, which is defined, for *nonsingular square matrices*, as

$$\kappa(\mathbf{A}) \equiv ||\mathbf{A}||||\mathbf{A}^{-1}|| \tag{3.38}$$

where $||\mathbf{A}||$ is a *norm* of a matrix $\mathbf{A}$.

Now, if the Euclidean norm is adopted for the condition number, then we have

$$\kappa \equiv \kappa_E = \sqrt{\frac{\lambda_l}{\lambda_s}} \equiv \sqrt{\frac{\lambda_l}{\lambda_s}} \tag{3.39}$$

in which λ_s is the smallest and λ_l is the largest eigenvalue of $\mathbf{A}\mathbf{A}^T$. It is now apparent that κ_E is bounded from below but unbounded from above:

$$\kappa_E \geq 1 \tag{3.40}$$

In fact, the above result holds for κ defined based on any norm. Moreover, if $\kappa(\mathbf{A})$ is based on the Frobenius norm $\|\mathbf{A}\|_F$, then

$$\begin{aligned}\kappa = \kappa_F(\mathbf{A}) &= \sqrt{\frac{1}{n}\mathrm{tr}(\mathbf{A}\mathbf{A}^T)}\sqrt{\frac{1}{n}\mathrm{tr}(\mathbf{A}^{-1}\mathbf{A}^{-T})} \\ &= \frac{1}{n}\sqrt{\mathrm{tr}(\mathbf{A}\mathbf{A}^T)\mathrm{tr}[(\mathbf{A}\mathbf{A}^T)^{-1}]} \equiv \frac{1}{n}\sqrt{\mathrm{tr}(\mathbf{A}^T\mathbf{A})\mathrm{tr}[(\mathbf{A}^T\mathbf{A})^{-1}]}\end{aligned} \tag{3.41}$$

Remarks:

- The condition number of a singular matrix is unbounded (tends to ∞)
- If a matrix $\mathbf{A}\mathbf{A}^T$ has all its eigenvalues identical, then $\mathbf{A}$ is said to be *isotropic*. Isotropic matrices have a $\kappa = 1$ *for κ defined in any matrix norm*. Isotropic matrices are optimally conditioned.

3.4.2 Gaussian Elimination

Various methods for computing a good approximation to *the* solution (3.35):

> *Iteratively*: Various types of methods, by the names Gauss-Jordan, Gauss-Seidel, successive-overrelaxation (SOR), etc. Used mainly for "large" systems (thousands of unknowns) that are *weakly coupled*; we will not handle such systems.
>
> *Symbolically*: Only possible for certain classes of $\mathbf{A}$ matrices, like tridiagonal, and for arbitrary matrices of modest size (n is below 5 or so.)
>
> *Gaussian elimination*, a.k.a. *LU-decomposition*: This is based on the observation that a *triangular system* is readily solved by either *backward* or *forward substitution*. $\mathbf{A}$ is decomposed into a *lower-triangular* and an *upper-triangular* factor, $\mathbf{L}$ and $\mathbf{U}$, respectively.

If $\mathbf{A}$ is nonsingular, but otherwise arbitrary, of $n \times n$, then, using **Gaussian elimination** we decompose $\mathbf{A}$ into

$$\mathbf{A} = \mathbf{L}\mathbf{U} \tag{3.42}$$

where $\mathbf{L}$ is *lower-triangular* and $\mathbf{U}$ is *upper-triangular*, namely,

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix} \tag{3.43}$$

$$\mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix} \tag{3.44}$$

Now, eq.(3.33) is rewritten as

$$\mathbf{L}\mathbf{U}\mathbf{x} = \mathbf{b} \quad \Rightarrow \quad \begin{cases} \mathbf{L}\mathbf{y} = \mathbf{b} \\ \mathbf{U}\mathbf{x} = \mathbf{y} \end{cases} \tag{3.45}$$

and hence, $\mathbf{x}$ is computed in two stages: First $\mathbf{y}$ is computed from a lower-triangular system; then, $\mathbf{x}$ is computed from an upper-triangular system. The lower-triangular system is solved for $\mathbf{y}$ by *forward substitution*; the upper-triangular system is solved for $\mathbf{x}$ by *backward substitution*. Note that

$$\det(\mathbf{A}) = \det(\mathbf{L})\det(\mathbf{U}) \tag{3.46a}$$

But, apparently,

$$\det(\mathbf{L}) = 1, \quad \det(\mathbf{U}) = \prod_1^n u_{ii} \quad \Rightarrow \quad \det(\mathbf{A}) = \det(\mathbf{U}) = \prod_1^n u_{ii} \tag{3.46b}$$

Hence, $\mathbf{A}$ is singular iff any of the diagonal entries of $\mathbf{U}$ vanishes.

3.4.3 Cholesky Decomposition

If $\mathbf{A}$ is *symmetric and positive-definite*, then it admits the **Cholesky decomposition**:

$$\mathbf{A} = \mathbf{L}^T\mathbf{L} \tag{3.47}$$

$$\mathbf{L} = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \tag{3.48}$$

where $\mathbf{L}$ is a *real*, lower-triangular matrix.

The solution of system (3.33) proceeds as in the general case, in two steps:

$$\mathbf{L}^T\mathbf{y} = \mathbf{b} \tag{3.49}$$

$$\mathbf{L}\mathbf{x} = \mathbf{y} \tag{3.50}$$

3.5 The Least-Square Solution of Overdetermined Linear Systems

We start with

Definition 3.5.1 A system of linear equations of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{3.51}$$

is *overdetermined* if $\mathbf{A}$ is rectangular, of $q \times n$, with $q > n$.

This means that the system has more equations than unknowns. In general, no $\mathbf{x}$ that verifies *all* the equations is available.

Definition 3.5.2 $\mathbf{A}$ is of full rank if its n $(< q)$ q-dimensional columns are linearly independent.

Remark: If $\mathbf{A}$ is of full rank, i.e., if rank $(\mathbf{A}) = n$, then

- The product $\mathbf{A}^T\mathbf{A}$ is nonsingular, and hence, positive-definite; moreover,
- as a consequence,

$$\det(\mathbf{A}^T\mathbf{A}) > 0 \tag{3.52}$$

For an arbitrary $\mathbf{x}$, there will be an error $\mathbf{e}$:

$$\mathbf{e} \equiv \mathbf{b} - \mathbf{A}\mathbf{x} \tag{3.53}$$

3.5.1 The Normal Equations

Problem: Find a particular $\mathbf{x}$, $\mathbf{x}_L$, that minimizes the Euclidean norm of the error, or its square, for that matter: $\|\mathbf{e}\|^2 = \mathbf{e}^T\mathbf{e}$.

Solution: Define the *objective function* f to be minimized as

$$f \equiv \frac{1}{2}\|\mathbf{e}\|^2 \quad \rightarrow \quad \min_{\mathbf{x}} \tag{3.54}$$

The *normality conditions* (NC) of Problem (3.54) are obtained upon zeroing the gradient of f with respect to $\mathbf{x}$:

$$\nabla f \equiv \frac{\partial f}{\partial \mathbf{x}} = \mathbf{0} \tag{3.55}$$

Moreover, ∇f is obtained from the "chain rule":

$$\frac{\partial f}{\partial x_i} = \frac{\partial e_j}{\partial x_i}\frac{\partial f}{\partial e_j}, \quad i = 1, \ldots, n$$

where the repeated index j indicates summation, for $j = 1, \ldots, q$. The foregoing relation can be written in compact form as

$$\nabla f \equiv \left(\frac{\partial \mathbf{e}}{\partial \mathbf{x}}\right)^T \frac{\partial f}{\partial \mathbf{e}} \tag{3.56}$$

Apparently, from the definitions of f and $\mathbf{e}$,

$$\frac{\partial \mathbf{e}}{\partial \mathbf{x}} = -\mathbf{A}, \quad \frac{\partial f}{\partial \mathbf{e}} = \mathbf{e} \equiv \mathbf{b} - \mathbf{A}\mathbf{x} \tag{3.57}$$

Upon plugging expressions (3.57) into eq.(3.55),

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b} \tag{3.58}$$

which is a system of n linear equations in n unknowns. This set of equations yields the NC of the problem at hand; the set is known as the *normal equations* of the given problem.

If $\mathbf{A}$ is of full-rank, then eq.(3.58) admits one *unique* solution—determined case—which is the *least-square solution* of the given system:

$$\mathbf{x}_L = \mathbf{A}^I\mathbf{b} \tag{3.59a}$$

with $\mathbf{A}^I$ defined as

$$\mathbf{A}^I \equiv (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T \tag{3.59b}$$

Here, $\mathbf{A}^I$ is termed the *left Moore-Penrose generalized inverse* (LMPGI) of the rectangular matrix $\mathbf{A}$.

Remarks:

- The condition number κ_E of the rectangular matrix $\mathbf{A}$ of $q \times n$, with $q > n$, based on the Euclidean norm, is defined in a similar way to that of a square matrix, with the difference that, in the case at hand, this is done in terms of the eigenvalues of $\mathbf{A}^T\mathbf{A}$;

- The condition number κ_E of $\mathbf{A}^T\mathbf{A}$ is the square root of the ratio of the largest to the smallest eigenvalues of $(\mathbf{A}^T\mathbf{A})(\mathbf{A}^T\mathbf{A})^T = (\mathbf{A}^T\mathbf{A})^2$;

- Hence, κ_E is given by the ratio of the largest to the smallest eigenvalues of $(\mathbf{A}^T\mathbf{A})$, i.e.,
$$\kappa_E(\mathbf{A}^T\mathbf{A}) = \kappa_E^2(\mathbf{A}) \tag{3.60}$$

- Thus, the roundoff-error amplification factor incurred in solving the normal equations (3.58) is the square of that incurred when "solving" eq.(3.33) in the determined case.

- Not only this. Formula (3.59a) is computationally expensive, for it involves:

  - the multiplication of $\mathbf{A}$ by its transpose from the left, which consumes n^2 scalar products of two q-dimensional vectors. Hence, $\mathbf{A}^T\mathbf{A}$ requires n^2q products and $n^2(q-1)$ additions;

  - the computation of the right-hand side of eq.(3.58), which entails, in turn, n scalar products of two q-dimensional vectors, i.e., $q \times n$ multiplications and $(q-1)n$ additions.

- In consequence,

  solving *numerically* normal equations should be avoided!

- In some cases, the normal equations allow for handling them with computer algebra, in which case roundoff-error amplification is not an issue. In these cases it is safe to work with these equations.

3.5.2 Householder Reflections

The good news is that there are alternatives to numerical normal-equation solving. One of these relies on *Householder reflections*, to be described presently.

Premultiply both sides of eq. (3.51) by n Householder reflections—$q \times q$ *improper orthogonal* matrices— $\mathbf{H}_i$, for $i = 1, \ldots, n$, i.e.,

$$\mathbf{HAx} = \mathbf{Hb} \tag{3.61}$$

where

$$\mathbf{H} = \mathbf{H}_n\mathbf{H}_{n-1}\ldots\mathbf{H}_1$$

The set $\{ \mathbf{H}_i \}_1^n$ is chosen so that

$$\mathbf{HA} = \begin{bmatrix} \mathbf{U} \\ \mathbf{O} \end{bmatrix}, \quad \mathbf{Hb} = \begin{bmatrix} \mathbf{b}_U \\ \mathbf{b}_L \end{bmatrix} \tag{3.62}$$

in which

- $\mathbf{U}$: a $n \times n$ *upper-triangular matrix*
- $\mathbf{O}$: the $(q-n) \times n$ zero matrix
- $\mathbf{b}_U$: a n-dimensional vector containing the upper n components of $\mathbf{Hb}$
- $\mathbf{b}_L$: a $(q-n)$-dimensional vector containing the lower $q-n$ components of $\mathbf{Hb}$

Thus, eq.(3.61) leads to two subsystems of equations:

$$\mathbf{Ux} = \mathbf{b}_U \tag{3.63a}$$

$$\mathbf{Ox} = \mathbf{b}_L \neq \mathbf{0} \tag{3.63b}$$

The least-square solution can be readily calculated by backward substitution from eq.(3.63a), and symbolically expressed as

$$\mathbf{x}_L = \mathbf{U}^{-1}\mathbf{b}_U. \tag{3.64}$$

Remark: Equation (3.63b) expresses a contradiction: The left-hand side is the product of the $(q-n) \times n$ zero matrix times the unknown vector; the right-hand side is not necessarily zero. Thus, eq.(3.63b) yields the least-square error associated with the solution $\mathbf{x}_L$: $\|\mathbf{b}_L\|$. Now we have an important result:

Theorem 3.5.1 (The Projection Theorem) *Let $\mathbf{e}_0$ denote the error vector of minimum Euclidean norm, i.e.,*

$$\mathbf{e}_0 \equiv \mathbf{b} - \mathbf{A}\mathbf{x}_L \tag{3.65}$$

Then, $\mathbf{e}_0$ is orthogonal *to the image of $\mathbf{x}_L$ under $\mathbf{A}$.*

Proof: We have

$$\mathbf{e}_0^T \mathbf{A}\mathbf{x}_L = (\mathbf{b} - \mathbf{A}\mathbf{x}_L)^T \mathbf{A}\mathbf{x}_L$$

Upon expansion,

$$\mathbf{e}_0^T \mathbf{A}\mathbf{x}_L = \mathbf{b}^T \mathbf{A}\mathbf{x}_L - \mathbf{x}_L^T \mathbf{A}^T \mathbf{A}\mathbf{x}_L$$

Plugging expressions (3.59a & b) into the above equation,

$$\begin{aligned}\mathbf{e}_0^T\mathbf{A}\mathbf{x}_L &= \mathbf{b}^T\mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} - \mathbf{b}^T\mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}\\ &= \mathbf{b}^T\mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} - \mathbf{b}^T\mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} = 0 \end{aligned} \tag{3.66}$$

thereby completing the proof. The Projection Theorem is illustrated in Fig. 3.1.
Remark: A $n \times n$ improper orthogonal matrix represents a *reflection*, i.e., a linear

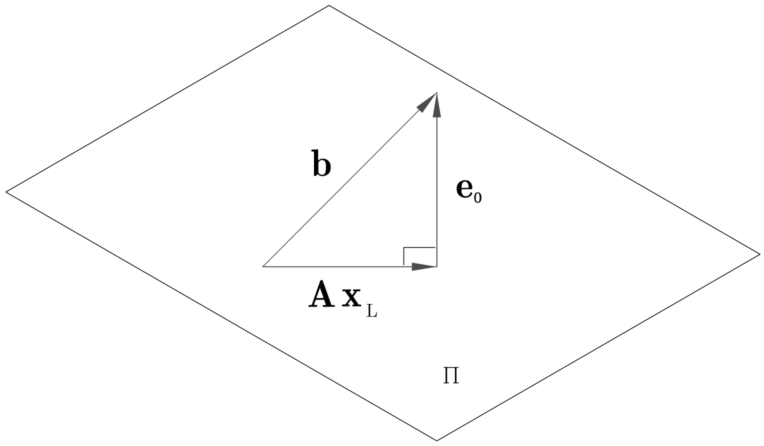


Figure 3.1: The Projection Theorem

transformation of a n-dimensional vector space that preserves both the *magnitude* of vectors—their Euclidean norm—and the inner product of any two vectors.

Problem: Find a linear transformation of the columns of the $q \times n$ matrix $\mathbf{A}$ that will render this matrix in upper-triangular form *without changing the geometric relations among the columns*, i.e., while preserving the inner products of any two of these columns, including the product of a column by itself.
Solution: Assume that we have applied reflections $\mathbf{H}_1$, $\mathbf{H}_2$, ..., $\mathbf{H}_{i-1}$, in this order, to $\mathbf{A}$ that have rendered its first $i-1$ columns in *upper-triangular form*, i.e.[2],

$$\mathbf{A}_{i-1} \equiv \mathbf{H}_{i-1}\ldots\mathbf{H}_2\mathbf{H}_1\mathbf{A}$$

[2]The entries of $\mathbf{A}_{i-1}$ are superscripted with an asterisk to distinguish them from the entries of the original $\mathbf{A}$.

$$
= \begin{bmatrix}
a_{11}^* & a_{12}^* & \cdots & a_{1,i-1}^* & a_{1i}^* & \cdots & a_{1n}^* \\
0 & a_{22}^* & \cdots & a_{2,i-1}^* & a_{2i}^* & \cdots & a_{2n}^* \\
0 & 0 & \cdots & a_{3,i-1}^* & a_{3i}^* & \cdots & a_{3n}^* \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & a_{i-1,i-1}^* & a_{i-1,i}^* & \cdots & a_{i-1,n}^* \\
0 & 0 & \cdots & 0 & a_{i,i}^* & \cdots & a_{i,n}^* \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & a_{q,i}^* & \cdots & a_{qn}^*
\end{bmatrix} \tag{3.67}
$$

The next Householder reflection, $\mathbf{H}_i$, is determined so as to render the last $q-i$ components of the ith column of $\mathbf{H}_i\mathbf{A}_{i-1}$ equal to zero, while leaving its first $i-1$ columns unchanged. We do this by setting

$$\alpha_i = \mathrm{sgn}(a_{ii}^*)\sqrt{(a_{ii}^*)^2 + (a_{i+1,i}^*)^2 + \cdots + (a_{qi}^*)^2} \tag{3.68}$$

$$\mathbf{u}_i = [\,0 \quad 0 \quad \cdots \quad 0 \quad a_{ii}^* + \alpha_i \quad a_{i+1,i}^* \quad \cdots \quad a_{qi}^*\,]^T \tag{3.69}$$

$$\mathbf{H}_i = \mathbf{1} - 2\frac{\mathbf{u}_i\mathbf{u}_i^T}{\|\mathbf{u}_i\|^2} \tag{3.70}$$

where the *signum* of x, $\mathrm{sgn}(x)$, is defined as $+1$ if $x > 0$, as -1 if $x < 0$, and is left undefined when $x = 0$.

Notice that

$$\frac{1}{2}\|\mathbf{u}_i\|^2 = \alpha_i(\mathbf{u}_i)_i = \alpha_i(a_{ii}^* + \alpha_i) \equiv \beta_i$$

and hence, the denominator appearing in the expression for $\mathbf{H}_i$ is calculated with one single addition and one single multiplication.

Exercise: Show that $\mathbf{H}_i\mathbf{H}_i^T = \mathbf{H}_i^T\mathbf{H}_i = \mathbf{1}$ and $\det(\mathbf{H}_i) = -1$.

Remark: $\mathbf{H}_i$ *reflects* vectors in q-dimensional space onto a hyperplane of unit normal $\mathbf{n} \equiv \mathbf{u}_i/\|\mathbf{u}_i\|$, as depicted in Fig. 3.2. It is noteworthy that

(a) α_i is defined with the sign of a_{ii}^* because β_i is a multiple of the ith component of $\mathbf{u}_i$, which is, in turn, the sum of a_{ii}^* and α_i, thereby guaranteeing that the absolute value of this sum will always be greater than the absolute value of each of its terms. If this provision were not made, then the resulting sum could be of a negligibly small absolute value, which would thus render β_i a very small positive number, thereby introducing unnecessarily an inadmissibly large roundoff-error amplification upon dividing the product $\mathbf{u}_i\mathbf{u}_i^T$ by β_i;

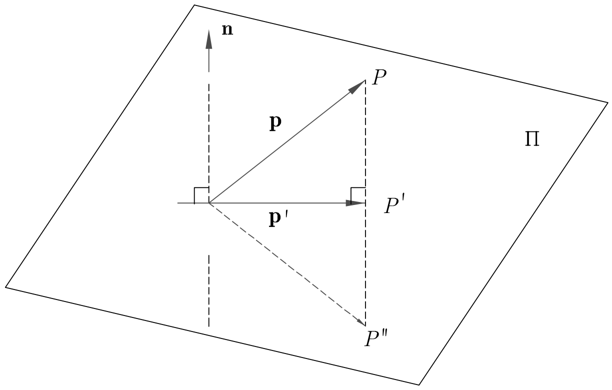


Figure 3.2: The geometric interpretation of the ith Householder reflection

(b) an arbitrary q-dimensional vector $\mathbf{v}$ is transformed by $\mathbf{H}_i$ with unusually few flops, namely,

$$\mathbf{H}_i\mathbf{v} = \mathbf{v} - \frac{1}{\beta_i}(\mathbf{v}^T\mathbf{u}_i)\mathbf{u}_i$$

Upon application of the n Householder reflections thus defined, the system at hand becomes

$$\mathbf{HAx} = \mathbf{Hb} \tag{3.71}$$

with $\mathbf{H}$ defined as

$$\mathbf{H} \equiv \mathbf{H}_n \ldots \mathbf{H}_2\mathbf{H}_1 \tag{3.72}$$

Notice that $\mathbf{HA}$ is in upper-triangular form. That is,

$$\mathbf{HA} = \begin{bmatrix} \mathbf{U} \\ \mathbf{O}_{q'n} \end{bmatrix}, \quad \mathbf{Hb} = \begin{bmatrix} \mathbf{b}_U \\ \mathbf{b}_L \end{bmatrix} \tag{3.73}$$

where: $q' \equiv q - n$; $\mathbf{O}_{q'n}$ is the $(q-n) \times n$ zero matrix; $\mathbf{b}_U$ is a n-dimensional vector; and $\mathbf{b}_L$ is a $(q-n)$-dimensional vector, normally different from zero.

The unknown $\mathbf{x}$ can thus be calculated from eq.(3.71) by back-substitution.

Remarks:

- The last m' components of the left-hand side of eq.(3.71) are zero.
- However, the corresponding components of the right-hand side of the same equation are not necessarily zero. What went wrong?
- Nothing! Recall that the overdetermined system (3.51) in general has no solution. The lower part of $\mathbf{b}$, $\mathbf{b}_L$, is then nothing but a q'-dimensional array

containing the nonzero components of the approximation error in the new coordinates. That is, the least-square error $\mathbf{e}_0$ in these coordinates, takes the form

$$\mathbf{e}_0 = \begin{bmatrix} \mathbf{0}_n \\ \mathbf{b}_L \end{bmatrix} \tag{3.74a}$$

Therefore,

$$\|\mathbf{e}_0\| = \|\mathbf{b}_L\| \tag{3.74b}$$

3.6 Nonlinear-Equation Solving: The Determined Case

Definition 3.6.1 A system of algebraic equations containing some that are not linear is termed *nonlinear*. If the number of equations is identical to the number of unknowns, the system is *determined*.

Example: Find the intersection of the circle and the hyperbola depicted in Fig. 3.3.
Solution: The equations of the circle and the hyperbola are

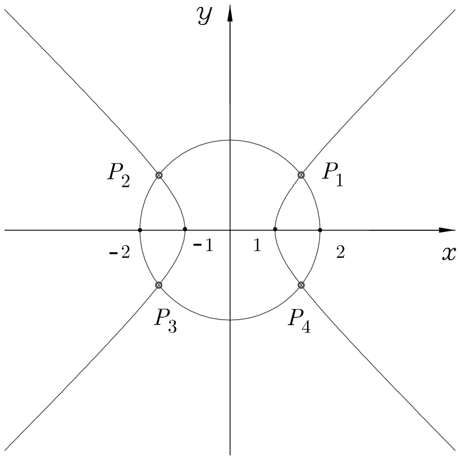


Figure 3.3: Intersection of a circle and a hyperbola

$$\phi_1(x, y) \equiv x^2 + y^2 - 4 = 0$$
$$\phi_2(x, y) \equiv x^2 - y^2 - 1 = 0$$

The solution to a nonlinear system of equations, when one exists at all, is usually *multiple*: The circle and the hyperbola of Fig. 3.3 intersect at four points $\{P_i\}_1^4$,

P_i	x_i	y_i
1	$\sqrt{5/2}$	$\sqrt{3/2}$
2	$\sqrt{5/2}$	$-\sqrt{3/2}$
3	$-\sqrt{5/2}$	$\sqrt{3/2}$
4	$-\sqrt{5/2}$	$-\sqrt{3/2}$

Table 3.1: The four intersection points of the circle and the hyperbola of Fig. 3.3

of coordinates (x_i, y_i), as displayed in Table 3.1. The problem may have **no real solution**, e.g., the circle and the hyperbola of Fig. 3.4 do not intersect. The system of equations from which the coordinates of the intersection points are to be computed is given below:

$$\phi_1(x,y) \equiv x^2 + y^2 - 1 = 0$$
$$\phi_2(x,y) \equiv x^2 - y^2 - 16 = 0$$

This system of equations admits no real solution!

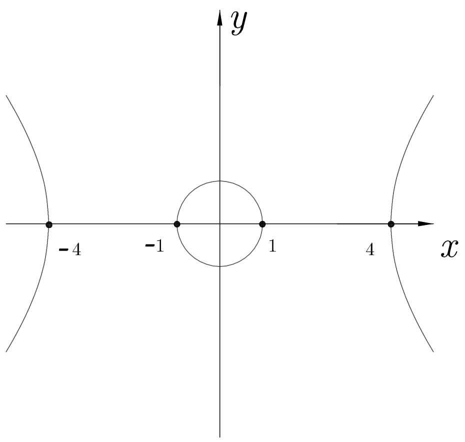


Figure 3.4: A circle and a hyperbola that do not intersect

In general, a determined nonlinear system of equations takes the form

$$\boldsymbol{\phi}(\mathbf{x}) = \mathbf{0} \tag{3.75}$$

where $\mathbf{x}$ and $\boldsymbol{\phi}$ are both n-dimensional vectors:

$$\mathbf{x} \equiv \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \boldsymbol{\phi} \equiv \begin{bmatrix} \phi_1(x_1, x_2, \ldots, x_n) \\ \phi_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ \phi_n(x_1, x_2, \ldots, x_n) \end{bmatrix} \tag{3.76}$$

3.6.1 The Newton-Raphson Method

We outline below the method of solution of determined nonlinear systems using the Newton-Raphson method. This is an *iterative method*, whereby a sequence of approximations is obtained that, if converging, it approaches the solution in a finite number of iterations within a prescribed *tolerance*.

A value $\mathbf{x}^0$ of $\mathbf{x}$ is given as an *initial guess*:

$$\mathbf{x}^0 \equiv [\, p_1 \quad p_2 \quad \ldots \quad p_n \,]^T$$

and $\boldsymbol{\phi}$ is evaluated at $\mathbf{x}^0$:

$$\boldsymbol{\phi}^0 \equiv \boldsymbol{\phi}(\mathbf{x}^0)$$

If the value $\mathbf{x}^0$ was chosen randomly, most likely it will not verify the given system of equations, i.e.,

$$\boldsymbol{\phi}^0 \neq \mathbf{0}$$

Next, we look for a "small" increment $\Delta\mathbf{x}$ of $\mathbf{x}$ (the increment is small if its norm—any norm—is small):

$$\Delta\mathbf{x} \equiv [\, \Delta x_1 \quad \Delta x_2 \quad \ldots \quad \Delta x_n \,]^T$$

Now, $\boldsymbol{\phi}(\mathbf{x}^0 + \Delta\mathbf{x})$ is evaluated up to its linear approximation (all quadratic and higher-order terms are dropped from its series expansion):

$$\boldsymbol{\phi}(\mathbf{x}^0 + \Delta\mathbf{x}) \approx \boldsymbol{\phi}(\mathbf{x}^0) + \left.\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^0} \Delta\mathbf{x} \tag{3.77}$$

The *Jacobian matrix* of $\boldsymbol{\phi}$ with respect to $\mathbf{x}$ is defined as the matrix of partial derivatives of the components of $\boldsymbol{\phi}$ with respect to all the components of $\mathbf{x}$:

$$\boldsymbol{\Phi} \equiv \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x}} = \begin{bmatrix} \partial\phi_1/\partial x_1 & \partial\phi_1/\partial x_2 & \cdots & \partial\phi_1/\partial x_n \\ \partial\phi_2/\partial x_1 & \partial\phi_2/\partial x_2 & \cdots & \partial\phi_2/\partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial\phi_n/\partial x_1 & \partial\phi_n/\partial x_2 & \cdots & \partial\phi_n/\partial x_n \end{bmatrix} \tag{3.78}$$

In the next step, we find $\Delta\mathbf{x}$ that renders zero the linear approximation of $\boldsymbol{\phi}(\mathbf{x}_0 + \Delta\mathbf{x})$:

$$\boldsymbol{\phi}_0 + \mathbf{\Phi}(\mathbf{x}^0)\Delta\mathbf{x} = \mathbf{0}$$

or

$$\mathbf{\Phi}(\mathbf{x}^0)\Delta\mathbf{x} = -\boldsymbol{\phi}^0 \tag{3.79}$$

whence $\Delta\mathbf{x}$ can be found using, for example, Gaussian elimination:

$$\Delta\mathbf{x} = -\mathbf{\Phi}_0^{-1}\boldsymbol{\phi}^0, \quad \mathbf{\Phi}_0 \equiv \mathbf{\Phi}(\mathbf{x}^0) \tag{3.80}$$

Next, $\mathbf{x}$ is updated:

$$\mathbf{x} \quad \leftarrow \quad \mathbf{x}^0 + \Delta\mathbf{x} \tag{3.81}$$

the procedure stopping when

$$\|\Delta\mathbf{x}\| \leq \epsilon_x \tag{3.82}$$

for a prescribed tolerance ϵ_x.

Remarks:

- Use the maximum norm to test convergence in eq.(3.82), for it costs virtually nothing;
- no guarantee that the Newton-Raphson method will converge at all;
- whether the Newton-Raphson method converges is dependent upon the initial guess, $\mathbf{x}^0$;
- the boundary between regions of convergence and divergence is a *fractal* (Mandelbrot, 1983; Gleick, 1988);
- when the Newton-Raphson method converges, it does so *quadratically*: At every iteration, *two* decimal places of accuracy are gained (Dahlquist and Björck, 1974).

3.7 Overdetermined Nonlinear Systems of Equations

A system of nonlinear equations of the form

$$\boldsymbol{\phi}(\mathbf{x}) = \mathbf{0} \tag{3.83}$$

where $\mathbf{x}$ is a n-dimensional vector and $\boldsymbol{\phi}$ is a q-dimensional vector, is *overdetermined* if $q > n$. Just as in the linear case, in general, no vector $\mathbf{x}$ can be found that verifies *all* the q scalar equations of the system. However, approximations can be found that minimize the least-square error of the approximation, as described in the balance of this Section. The method of solution adopted here is the overdetermined counterpart of the Newton-Raphson method.

3.7.1 The Newton-Gauss Method

Problem: Find an *approximate solution* to system (3.83) that verifies those equations with the *least-square error*:

$$f(\mathbf{x}) = \frac{1}{2}\boldsymbol{\phi}^T\mathbf{W}\boldsymbol{\phi} \quad \rightarrow \quad \min_{\mathbf{x}} \tag{3.84}$$

where $\mathbf{W}$ is a $q \times q$ positive-definite *weighting matrix*.

Solution: We follow a procedure similar to Newton-Raphson's, which is known as the *Newton-Gauss method*, as described below:

First, an initial guess $\mathbf{x}^0$ of $\mathbf{x}$ is given; then, we produce the sequence

$$\mathbf{x}^1,\, \mathbf{x}^2,\, \ldots, \tag{3.85}$$

such that

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k \tag{3.86}$$

Calculation of $\Delta\mathbf{x}^k$:

- Factor $\mathbf{W}$ into its two Cholesky factors:

  $$\mathbf{W} = \mathbf{V}^T\mathbf{V} \tag{3.87}$$

  which is possible because $\mathbf{W}$ is assumed positive-definite.

- Compute $\Delta\mathbf{x}^k$ as the *least-square solution* of the unconstrained overdetermined linear system

  $$\mathbf{V}\boldsymbol{\Phi}(\mathbf{x}^k)\Delta\mathbf{x}^k = -\mathbf{V}\boldsymbol{\phi}(\Delta\mathbf{x}^k) \tag{3.88}$$

  with $\boldsymbol{\Phi}(\mathbf{x})$ defined as the $q \times n$ Jacobian matrix of the vector function $\boldsymbol{\phi}(\mathbf{x})$, i.e.,

  $$\boldsymbol{\Phi}(\mathbf{x}) = \frac{\partial\boldsymbol{\phi}(\mathbf{x})}{\partial\mathbf{x}} \tag{3.89}$$

  Drop superscripts for the sake of notation-simplicity and recall eqs.(3.59a & b):

  $$\Delta\mathbf{x} = -(\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\phi} \tag{3.90}$$

  This procedure is iterative, stopping when a *convergence criterion is met.*

The Damping Factor

When implementing the Newton-Gauss method, the objective function f may increase upon correcting $\mathbf{x}^k$ according to eq.(3.86), i.e.

$$f(\mathbf{x}^{k+1}) > f(\mathbf{x}^k) \tag{3.91}$$

This increase gives rise to oscillations and sometimes even leads to divergence. One way to cope with this situation is by introducing *damping*. Instead of using the whole increment $\Delta\mathbf{x}^k$, we use a fraction of it, i.e.

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha\Delta\mathbf{x}^k, \quad 0 < \alpha < 1 \tag{3.92}$$

where α is known as the *damping factor*.

3.7.2 Convergence Criterion

Calculate first $\nabla f(\mathbf{x})$:

$$\nabla f(\mathbf{x}) \equiv \frac{\partial f}{\partial \mathbf{x}} = \left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x}}\right)^T \frac{\partial f}{\partial \boldsymbol{\phi}} \tag{3.93}$$

$$\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x}} \equiv \mathbf{\Phi}, \quad \frac{\partial f}{\partial \boldsymbol{\phi}} = \mathbf{W}\boldsymbol{\phi} \tag{3.94}$$

Hence, the condition for a stationary point is

$$\mathbf{\Phi}^T\mathbf{W}\boldsymbol{\phi} = \mathbf{0} \tag{3.95}$$

which is the *normality condition* of eq.(3.84).

It is thus apparent that, at a stationary point of f, $\boldsymbol{\phi}(\mathbf{x})$ **need not vanish**, as is the case of unconstrained optimization, to be discussed in Chapter 4;however, $\boldsymbol{\phi}(\mathbf{x})$ **must lie in the nullspace of $\mathbf{\Phi}^T\mathbf{W}$**. Moreover, from eqs.(3.90) and (3.95) follows that, at a stationary point, $\Delta\mathbf{x}$ vanishes. Hence, the convergence criterion is

$$||\Delta\mathbf{x}|| < \epsilon \tag{3.96}$$

where ϵ is a prescribed tolerance.

Remarks:

- The normality condition (3.95) alone does not guarantee a minimum, but only a *stationary point*.

- However, as it turns out, if the procedure converges, then it does so, to a second-order approximation, to a minimum, and neither to a maximum nor a to saddle point, as we prove below.

The sequence $f(\mathbf{x}^0)$, $f(\mathbf{x}^1)$, ..., $f(\mathbf{x}^k)$, $f(\mathbf{x}^{k+1})$, ..., obtained from the sequence of $\mathbf{x}$ values, evolves, to a first order, as $\Delta f(\mathbf{x})$, given by

$$\Delta f = \left(\frac{\partial f}{\partial \mathbf{x}}\right)^T \Delta \mathbf{x} \tag{3.97}$$

i.e.,

$$\Delta f = \boldsymbol{\phi}^T \mathbf{W} \boldsymbol{\Phi} \Delta \mathbf{x} \tag{3.98}$$

Upon plugging expression (3.90) of $\Delta \mathbf{x}$ into eq. (3.98), we obtain

$$\Delta f = -\boldsymbol{\phi}^T \underbrace{\mathbf{W}\boldsymbol{\Phi}(\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{W}}_{\mathbf{M}}\,\boldsymbol{\phi} = -\boldsymbol{\phi}^T\mathbf{M}\boldsymbol{\phi} \tag{3.99}$$

where, apparently, $\mathbf{M}$ is a $q \times q$ positive-definite matrix. As a consequence, $\boldsymbol{\phi}^T\mathbf{M}\boldsymbol{\phi}$ becomes a positive-definite quadratic expression of $\boldsymbol{\phi}$; hence, Δf is negative definite. Thus, the second-order approximation of $f(\mathbf{x})$ is negative-definite, and hence, the sequence of f values *decreases monotonically*. That is, in the neighbourhood of a stationary point the first-order approximation of $\boldsymbol{\phi}(\mathbf{x})$ is good enough, and hence, if the procedure **converges**, it does so to a **minimum**.

The reader may wonder whether the Newton-Raphson method can be used to solve nonlinear least-square problems. Although the answer is *yes*, the Newton-Raphson method is not advisible in this case, as made apparent below.

Recall ∇f from eqs.(3.84) and (3.85):

$$\nabla f(\mathbf{x}) = \frac{\partial f}{\partial \mathbf{x}} = \underbrace{\boldsymbol{\Phi}^T(\mathbf{x})}_{n\times q}\underbrace{\mathbf{W}}_{q\times q}\underbrace{\boldsymbol{\phi}(\mathbf{x})}_{q-\mathrm{dim}}$$

$$\nabla f(\mathbf{x}) = \mathbf{0} \;\Rightarrow\; \underbrace{\boldsymbol{\Phi}^T(\mathbf{x})\mathbf{W}\boldsymbol{\phi}(\mathbf{x})}_{\equiv\boldsymbol{\psi}(\mathbf{x})\in\mathbb{R}^n} = \mathbf{0} \tag{NC}$$

thereby obtaining a determined system of n equations in n unknowns. This system can be solved using Newton-Raphson method which requires $\nabla\boldsymbol{\psi}(\mathbf{x})$:

$$\nabla\boldsymbol{\psi}(\mathbf{x} = \frac{\partial \boldsymbol{\psi}}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}}[\,\underbrace{\boldsymbol{\Phi}^T(\mathbf{x})}_{(\partial\boldsymbol{\phi}/\partial\mathbf{x})^T}\;\mathbf{W}\boldsymbol{\phi}(\mathbf{x})]$$

That is, $\nabla\boldsymbol{\psi}(\mathbf{x})$ involves second-order derivatives of $\boldsymbol{\psi}$ with respect to $\mathbf{x}$:

$$\frac{\partial^2 \boldsymbol{\phi}_i}{\partial x_j \partial x_i}, \ i = 1, \ldots, n$$

In summary, the Newton-Raphson method is too cumbersome and prone to ill-conditioning, for it is based on the normality conditions of the problem at hand.

3.8 Computer Implementation Using ODA—C-Library of Routines for Optimum Design

ODA is a C library of subroutines for optimization problems. The source file of this package, implemented in C, consists of a number of subroutines designed and classified based on their application. At the beginning of each subroutine a detailed description of the purpose and usage of the subroutine is included. Moreover, data validation has been considered in the software. In order to solve a problem, the user simply calls one corresponding C subroutine.

Since the solutions for linear problems are *direct*—as opposed to *iterative*—the use of ODA to solve linear problems requires only information on the problem parameters, such as matrices $\mathbf{A}$, $\mathbf{C}$, and $\mathbf{W}$, as well as vectors $\mathbf{b}$ and $\mathbf{d}$, as applicable. For nonlinear problems, the solution is iterative, and hence, the user is required to provide functions describing $\boldsymbol{\phi}(\mathbf{x})$, $\mathbf{h}(\mathbf{x})$, $\boldsymbol{\Phi}(\mathbf{x})$, $\mathbf{J}(\mathbf{x})$, as needed. These functions are provided via subroutines in forms that can be called by the package. In addition to this information, the user is also required to provide an initial guess $\mathbf{x}_0$ of $\mathbf{x}$, so that the iterative procedure can be started.

1. **Unconstrained linear problems:** Subroutine `MNSLS` is used to find the minimum-norm solution of an underdetermined linear system, while subroutine `LSSLS` is used to find the least-square approximation of an overdetermined linear system. `LSSLS` can also handle determined systems, i.e., systems of as many equations as unknowns.

2. **Unconstrained nonlinear problems:** Subroutine `LSSNLS` is used to solve this type of problems. Since the nonlinear functions and their associated gradient matrices are problem-dependent, the user is required to provide two subroutines that are used to evaluate the foregoing items, namely,

   - `FUNPHI`: This subroutine is used to evaluate the q-dimensional vector function $\boldsymbol{\phi}(\mathbf{x})$ in terms of the given n-dimensional vector $\mathbf{x}$.

- DPHIDX: This subroutine is used to evaluate the $q \times n$ gradient matrix $\mathbf{\Phi}$ of the vector-function $\boldsymbol{\phi}(\mathbf{x})$ with respect to $\mathbf{x}$, at the current value of $\mathbf{x}$. Moreover, an initial guess of $\mathbf{x}$ is required when calling this subroutine.

3. **Constrained linear problems:** Subroutine LSSCLS is used to solve this type of problems.

4. **Constrained nonlinear problems:** Subroutine LSSCNL is used for solving this type of problems. Before calling LSSCNL, the user is required to provide four problem-dependent subroutines: Two of these are FUNPHI and DPHIDX, already derscribed in item 2 above. The other two are used to evaluate the left-hand sides of the constraint equations and their gradient matrix, as listed below:

   - FUNH: This subroutine is used to evaluate the l-dimensional constraint function $\mathbf{h}$ in terms of the given n-dimensional vector $\mathbf{x}$.

   - DHDX: This subroutine is used to evaluate the $l \times n$ gradient matrix $\mathbf{J}$ of the vector-function $\mathbf{h}(\mathbf{x})$ in terms of the given n-dimensional vector $\mathbf{x}$. Moreover, an initial guess of $\mathbf{x}$ is required when calling LSSCNL.

5. **Constrained problems with arbitrary objective function:** Subroutine ARBITRARY is used for solving this type of problems. Before calling ARBITRARY, the user is required to provide four problem-dependent subroutines: Two of these are FUNPHI and DPHIDX, as described in item 2 above. The other two subroutines are used to evaluate the left-hand sides of the constraint equations and their gradient matrix, as listed below:

   - phi: Subroutine used to evaluate the objective function $\boldsymbol{\phi}(\mathbf{x})$ in terms of the given n-dimensional vector $\mathbf{x}$.

   - h: Subroutine used to evaluate the l-dimensional constraint function $\mathbf{h}$ in terms of the given n-dimensional vector $\mathbf{x}$.

   - J: Subroutine used to evaluate the $l \times n$ gradient matrix $\mathbf{J}$ of the vector-function $\mathbf{h}(\mathbf{x})$ at the current value of $\mathbf{x}$.

   - gradient: Subroutine used to evaluate the n-dimensional gradient ∇f of the objective function $f(\mathbf{x})$ at the current value of vector $\mathbf{x}$.

- `Hessian`: Subroutine used to evaluate the $n \times n$ Hessian matrix $\nabla\nabla f$ of the objective function $f(\mathbf{x})$ at the current value of vector $\mathbf{x}$. Moreover, an initial guess of $\mathbf{x}$ is required when calling `ARBITRARY`.

Chapter 4

Unconstrained Optimization

4.1 Introduction

We start by studying the simplest problem in multivariable optimization, namely, the *unconstrained minimization* of a *smooth* scalar objective function $f(\mathbf{x})$ of the n-dimensional *design-variable vector*, or *design vector* (DV) for brevity, that we denote by $\mathbf{x}$. The main result here is the *normality conditions* (NC) of the problem at hand. We derive the *first-order* (FONC), which are *necessary* for a stationary point (SP); then, we derive the *second-order* (SONC), which are *sufficient* for a *minimum*, a *maximum* or a *saddle point*. These three kinds of SP are duly characterized.

4.2 The Normality Conditions

Under the smoothness assumption, the objective function is *continuous* and has *continuous first- and second-order derivatives*. The problem at hand is, moreover,

$$f(\mathbf{x}) \quad \rightarrow \quad \min_{\mathbf{x}} \tag{4.1}$$

Since the problem under study is unconstrained, the search of the minimum is conducted over the whole design space $\mathbb{R}^n$, which eases the search tremendously. Notice that every point of the design space is characterized by a position vector $\mathbf{x}$, which defines a *design*, and hence, every such point represents one design. For conciseness, we will refer to a point and the design that the point represents by its position vector.

Now, for $f(\mathbf{x})$ to attain a minimum at a certain point $\mathbf{x}_o$ of the design space, the point must be, first and foremost, *stationary*, i.e., the *gradient* ∇f of the objective function with respect to the design vector must vanish:

$$\nabla f \equiv \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}_o} = \mathbf{0} \tag{4.2a}$$

which is known as the *first-order normality condition.* As a matter of fact, the above relation is short-hand for n normality conditions, one for each component of the ∇f vector, namely,

$$\nabla f \equiv \frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \\ \vdots \\ \partial f/\partial x_n \end{bmatrix} \tag{4.2b}$$

However, a stationary point can be a *minimum*, a *maximum* or a *saddle point*, to a second-order approximation. To characterize each case, we expand, to this order of approximation, $f(\mathbf{x})$ around $\mathbf{x} = \mathbf{x}_o$:

$$f(\mathbf{x}) = f(\mathbf{x}_o) + \nabla f\,|_{\mathbf{x}_o}\,(\mathbf{x} - \mathbf{x}_o) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_o)^T \nabla\nabla f\,|_{\mathbf{x}_o}(\mathbf{x} - \mathbf{x}_o) + \mathrm{HOT} \tag{4.3a}$$

where HOT stands for "higher-order-terms", while $\nabla\nabla f$, the *Hessian* of f with respect to $\mathbf{x}$, is a matrix of second derivatives, namely,

$$\nabla\nabla f \equiv \frac{\partial^2 f}{\partial \mathbf{x}^2} = \begin{bmatrix} \partial^2 f/\partial x_1^2 & \partial^2 f/\partial x_1 \partial x_2 & \cdots & \partial^2 f/\partial x_1 \partial x_n \\ \partial^2 f/\partial x_2 \partial x_1 & \partial^2 f/\partial x_2^2 & \cdots & \partial^2 f/\partial x_2 \partial x_n \\ \vdots & \vdots & \ddots & \vdots \\ \partial^2 f/\partial x_n \partial x_1 & \partial^2 f/\partial x_n \partial x_2 & \cdots & \partial^2 f/\partial x_n^2 \end{bmatrix} \tag{4.3b}$$

Notice that, by virtue of the smoothness assumption,

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}, \quad \text{for} \quad i, j = 1, 2, \ldots, n \tag{4.4}$$

which follows after *Schwartz's Theorem*: *Given a continuous function $f(\mathbf{x})$ with first- and second-order continuous derivatives, the order of differentiation in computing the second derivatives is immaterial.*

As a consequence of eq.(4.4), then,

- The Hessian of f with respect to $\mathbf{x}$ is a symmetric $n \times n$ matrix, and
- the eigenvalues of the Hessian matrix are all real and its eigenvectors are mutually orthogonal.

At a stationary point $\mathbf{x}_o$, then, and up to a second-order approximation, eq.(4.3a) leads to

$$\Delta f \equiv f(\mathbf{x}) - f(\mathbf{x}_o) \approx \frac{1}{2}(\mathbf{x} - \mathbf{x}_o)^T \nabla\nabla f \,|_{\mathbf{x}_o} (\mathbf{x} - \mathbf{x}_o) \tag{4.5}$$

Now we have that

- If, for any $\Delta\mathbf{x} \equiv \mathbf{x} - \mathbf{x}_o$, $\Delta f(\mathbf{x}) > 0$, then the stationary point (SP) $\mathbf{x}_o$ is a *local minimum* of $f(\mathbf{x})$;
- if, for any $\Delta\mathbf{x} \equiv \mathbf{x} - \mathbf{x}_o$, $\Delta f(\mathbf{x}) < 0$, then the SP $\mathbf{x}_o$ is a *local maximum* of $f(\mathbf{x})$; and
- otherwise, the SP $\mathbf{x}_o$ is a *saddle point*.

It is not practical to test a stationary point for the sign of Δf for every possible $\Delta\mathbf{x}$. However, it is possible to characterize the nature of the stationary point $\mathbf{x}_o$ by means of the signs of the eigenvalues of the Hessian matrix. To this end, we recall the characterization of positive-definite, positive-semidefinite and sign-indefinite matrices given above. In this light, then,

- the stationary point $\mathbf{x}_o$ is a *local minimum* if the Hessian evaluated at this point is positive-definite;
- the SP is a *local maximum* if the Hessian evaluated at this point is negative-definite;
- the SP is a *saddle point* if the Hessian evaluated at this point is sign-indefinite.

4.3 Methods of Solution

The variety of methods available is immensely rich. In a nutshell, the various methods can be classified according to one criterion: the requirement of partial derivatives of the objective function. We thus have:

(*i*) **Direct methods:** No derivatives are required.

(*ii*) **Gradient Methods:** Only first-order derivatives of the objective function with respect to all design variables are required.

(*iii*) **Newton methods:** First- and second-order derivatives of the objective function with respect to all all design variables are required.

Needless to say, direct methods are the most general—the least demanding—and simplest to implement, the price to be paid for the lack of information on derivatives being the speed of convergence. These methods are the slowest to converge. Gradient methods are faster, with a *linear convergence rate*, which means that the error between the current *iterate* $\mathbf{x}^k$ and *the closest local minimum* decreases by one order of magnitude at each iteration. The Newton-Raphson method, and variations thereof, generically termed *Newton methods*, resort to first- and second-order derivatives. Newton methods converge *quadratically*, which means that the aforementioned error decreases by two orders of magnitude per iteration.

4.4 Direct Methods

Direct methods are based on function evaluations and nothing else. While these methods are slow to converge, they can handle discontinuous functions. There are various of these: random (random jumps, random walks); Hooke and Jeeves; Powell; and the simplex (Nelder-Mead) method. We outline the last three of these methods below.

The main concept behind direct methods is the *pattern directions*, namely, the directions of search, along which the minimum is approached. These three methods differ on the way of defining the pattern directions.

4.4.1 The Hooke and Jeeves Method

In this method, the search directions are fixed. The method thus starts by defining a set of unit vectors $\{\mathbf{u}_i\}_1^n$ in the directions of the n design variables, along which the search is conducted.

Hooke & Jeeves Algorithm

1. `Define starting base point` $\mathbf{x}^1$ `and prescribed length` $\Delta\mathbf{x}$ `of search step`
2. $f_k \leftarrow f(\mathbf{x}^k)$; $1 \leftarrow i$; $\mathbf{y}^{k,0} \leftarrow \mathbf{x}^k$
3. `For i = 1 to n do`

$$f^+ \leftarrow f(\mathbf{y}^{k,i-1} + \mathbf{u}_i \Delta x_i)$$
$$f^- \leftarrow f(\mathbf{y}^{k,i-1} - \mathbf{u}_i \Delta x_i)$$
`if` $f^+ < f(\mathbf{y}^{k,i-1})$ `then` $\mathbf{y}^{k,i} \leftarrow \mathbf{y}^{k,i-1} + \mathbf{u}_i \Delta x_i$

if $f^- < f(\mathbf{y}^{k,i-1})$ then $\mathbf{y}^{k,i} \leftarrow \mathbf{y}^{k,i-1} - \mathbf{u}_i \Delta x_i$

else $\mathbf{y}^{k,i} \leftarrow \mathbf{y}^{k,i-1}$

enddo

4. if $\mathbf{y}^{k,n} = \mathbf{x}^k$ then $\Delta\mathbf{x} \leftarrow \frac{1}{2}\Delta\mathbf{x}$ go to 3

   else $\mathbf{x}^{k+1} \leftarrow \mathbf{y}^{k,n}$

5. $\mathbf{s} \leftarrow \mathbf{x}^{k+1} - \mathbf{x}^k$ % pattern direction

   Find λ that minimizes $f(\mathbf{x}^{k+1} + \lambda\mathbf{s})$

   $\mathbf{y}^{k+1,n} + \lambda\mathbf{s}$

6. $k \leftarrow k+1,\ f_k \leftarrow f(\mathbf{y}^{k,0}), 1 \leftarrow i$

   repeat 3

   if $f(\mathbf{y}^{k,n}) < f(\mathbf{x}^k)$ then $\mathbf{x}^{k+1} \leftarrow \mathbf{y}^{k,n}$

   else $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k;\ \Delta\mathbf{x} \leftarrow \frac{1}{2}\Delta\mathbf{x}$ go to 2

7. if $\|\Delta\mathbf{x}\| < \epsilon$ then stop

   else go to 2

4.4.2 The Powell Method (Conjugate Directions)

Let $\mathbf{A} = \mathbf{A}^T \in \mathbb{R}^{n\times n}$ be positive-definite. Vectors $\mathbf{x}$, $\mathbf{y} \in \mathbb{R}^n$ are said to be $\mathbf{A}$-conjugate if

$$\mathbf{x}^T\mathbf{A}\mathbf{y} = \mathbf{y}^T\mathbf{A}\mathbf{x} = 0 \tag{4.6}$$

In the area of mechanical systems under linear vibrations, for n degrees of freedom, the ith and jth modal vectors $\mathbf{u}^i$ and $\mathbf{u}^j$ are $\mathbf{K}$- and $\mathbf{M}$-conjugate, i.e.,

$$(\mathbf{u}^i)^T\mathbf{K}\mathbf{u}^j = 0, \quad \text{and } (\mathbf{u}^i)^T\mathbf{M}\mathbf{u}^j = 0$$

where $\mathbf{K}$ and $\mathbf{M}$ are the $n \times n$ positive-definite stiffness and mass matrices, respectively.

Theorem 4.4.1 *(cf. Rao's Theorem 6.1)*
Let $\mathbf{x} \in \mathbb{R}^n$ *and*

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c, \quad \mathbf{A} = \mathbf{A}^T > \mathbf{O}$$

Further, let Π_1 and Π_2 be two hyperplanes in $\mathbb{R}^n$ parallel to each other. If $\mathbf{x}^i$ is the minimum of $f(\mathbf{x})$, with $\mathbf{x}^i \in \Pi_i$, $i = 1, 2$, then $\mathbf{x}^2 - \mathbf{x}^1$ is $\mathbf{A}$-conjugate with any vector $\mathbf{y} \in \mathbb{R}^n$ lying in a hyperplane normal to Π_1 and Π_2, i.e.,

$$(\mathbf{x}^2 - \mathbf{x}^1)^T \mathbf{A}\mathbf{y} = 0$$

Proof: Π_1 and Π_2 are defined by

$$\Pi_i: \ \mathbf{C}\mathbf{x} = \mathbf{d}^i, \ i = 1, 2; \quad \mathbf{C} \in \mathbb{R}^{p \times n}, \ p < n \tag{4.7}$$

The algebraic interpretation of $\mathbf{y}$ lying in a plane perpendicular to Π_1 and Π_2 is that $\mathbf{y} \in \mathcal{R}\ (\mathbf{C}^T)$, i.e., there exists a $\mathbf{v} \in \mathbb{R}^p$ such that

$$\mathbf{y} = \mathbf{C}^T\mathbf{v}$$

and hence, if $\mathbf{L}$ is an *orthogonal complement* of $\mathbf{C}$, i.e., if

$$\underbrace{\mathbf{C}}_{p\times n}\underbrace{\mathbf{L}}_{n\times n'} = \underbrace{\mathbf{O}}_{p\times n'}, \quad n' \equiv n - p$$

then we can write

$$\mathbf{y} = \mathbf{L}\mathbf{u}, \quad \mathbf{u} \in \mathbb{R}^{n-p}$$

Now, $\mathbf{x}^i$ is found as the solution to

$$\min_{\mathbf{x}} f(\mathbf{x})$$

subject to eq.(4.7). We solve the foregoing problem as an unconstrained problem by means of the Lagrangian[1]

$$F_i \equiv f(\mathbf{x}) + (\boldsymbol{\lambda}^i)^T(\mathbf{C}\mathbf{x} + \mathbf{d}^i) \to \min_{\mathbf{x}}, \quad i = 1, 2$$

subject to no constraints, and denote the solution $\mathbf{x}^i$. The normality conditions of the foregoing problem are, for $i = 1, 2$,

$$\nabla F_i = \mathbf{A}\mathbf{x} + \mathbf{b} + \mathbf{C}^T\boldsymbol{\lambda}^i = \mathbf{0}_n, \tag{4.8a}$$

$$\mathbf{C}\mathbf{x} - \mathbf{d}^i = \mathbf{0}_p \tag{4.8b}$$

which yield a system of $p + n$ equations for the $p + n$ unknowns $\mathbf{x}$ and $\boldsymbol{\lambda}^i$. Upon solving for $\mathbf{x} = \mathbf{x}^i$ from eq.(4.8a), for $i = 1, 2$, we obtain

$$\mathbf{x}^i = -\mathbf{A}^T(\mathbf{C}^T\boldsymbol{\lambda}^i + \mathbf{b}) \tag{4.9}$$

[1] Details on constrained optimization are given in Chapter 5.

Substitution of the foregoing expression into eq.(4.7) yields, always for $i = 1, 2$,

$$-\mathbf{C}\mathbf{A}^{-1}(\mathbf{C}^T\boldsymbol{\lambda}^i + \mathbf{b}) = \mathbf{d}^i$$

or

$$\mathbf{C}\mathbf{A}^{-1}\mathbf{C}^T\boldsymbol{\lambda}^i = -\mathbf{C}\mathbf{A}^{-1}\mathbf{b} - \mathbf{d}^i$$

Hence,

$$\boldsymbol{\lambda}^i = -(\mathbf{C}\mathbf{A}^{-1}\mathbf{C}^T)^{-1}(\mathbf{C}\mathbf{A}^{-1}\mathbf{b} + \mathbf{d}^i) \tag{4.10}$$

Substitution of eq.(4.10) into eq.(4.9) leads to

$$\begin{aligned}\mathbf{x}^i &= \mathbf{A}^{-1}[\mathbf{C}^T(\mathbf{C}\mathbf{A}^{-1}\mathbf{C}^T)^{-1}(\mathbf{C}\mathbf{A}^{-1}\mathbf{b} + \mathbf{d}^i) + \mathbf{b}] \\ &= \mathbf{A}^{-1}[\mathbf{C}^T(\mathbf{C}\mathbf{A}^{-1}\mathbf{C}^T)^{-1}\mathbf{C}\mathbf{A}^{-1} + \mathbf{1}]\mathbf{b} + \mathbf{A}^{-1}[\mathbf{C}^T(\mathbf{C}\mathbf{A}^{-1}\mathbf{C}^T)^{-1}\mathbf{d}^i\end{aligned}$$

Therefore,

$$\mathbf{x}^2 - \mathbf{x}^1 = \mathbf{A}^{-1}\mathbf{C}^T(\mathbf{C}\mathbf{A}^{-1}\mathbf{C}^T)^{-1}(\mathbf{d}^2 - \mathbf{d}^1)$$

Hence,

$$(\mathbf{x}^2 - \mathbf{x}^1)^T\mathbf{A}\mathbf{y} \equiv (\mathbf{x}^2 - \mathbf{x}^1)^T\mathbf{A}\mathbf{L}\mathbf{u} = (\mathbf{d}^2 - \mathbf{d}^1)^T(\mathbf{C}\mathbf{A}^{-1}\mathbf{C}^T)^{-1}\underbrace{\mathbf{C}\,\underbrace{\mathbf{A}^{-1}\mathbf{A}}_{\mathbf{1}}\,\mathbf{L}}_{\mathbf{CL}=\mathbf{O}_{pn'}}\mathbf{u}$$

That is,

$$(\mathbf{x}^2 - \mathbf{x}^1)^T\mathbf{A}\mathbf{y} = 0$$

thereby completing the proof.

The Powell Algorithm

Data: A set of linearly independent directions $\mathcal{D} = \{\boldsymbol{\xi}^i\}^n$ and an initial guess $\mathbf{x}^0$

1. `For` $k = 1$ `to` n `do`
   `find minimizer` λ_k `of` $f(\mathbf{x}^{k-1} + \lambda_k\boldsymbol{\xi}^k)$
   $\mathbf{x}^k \leftarrow \mathbf{x}^{k-1} + \lambda_k\boldsymbol{\xi}^k$
   `enddo`
2. `Find` $m \in \{1, \ldots, n\}$ `such that` $f(\mathbf{x}^{m-1}) - f(\mathbf{x}^m)$ `is a maximum; then` $\Delta \leftarrow f(\mathbf{x}^{m-1}) - f(\mathbf{x}^m)$
3. $f_1 \leftarrow f(\mathbf{x}^0),\ f_2 \leftarrow f(\mathbf{x}^n),\ f_3 \leftarrow f(2\mathbf{x}^n - \mathbf{x}^0)$
4. `if` $\left\{\begin{array}{c} f_3 \geq f_1, \text{ or} \\ (f_1 - 2f_2 + f_3)(f_1 - f_2)^2 \geq \frac{1}{2}(f_1 - f_3)^2\Delta \end{array}\right.$
   `then keep` $\mathcal{D}$`; else`

5. $\boldsymbol{\xi} \leftarrow \mathbf{x}^n - \mathbf{x}^0$
   find λ that minimizes $f(\mathbf{x}^n + \lambda\boldsymbol{\xi})$
   $\mathcal{D} \leftarrow \{\boldsymbol{\xi}^1, \ldots, \boldsymbol{\xi}^{m-1}, \boldsymbol{\xi}^{m+1}, \ldots, \boldsymbol{\xi}^n, \boldsymbol{\xi}\}$
   $\mathbf{x}^0 \leftarrow \mathbf{x}^n + \lambda\boldsymbol{\xi}$
   if $\mathbf{x}^0$ is a minimum, stop; else, go to 1.

Note: As initial $\mathcal{D}$, use the coordinate axes.

Theorem 4.4.2 *For quadratic objective functions, Powell's directions are* $(\nabla\nabla f)$-*conjugate.*

4.4.3 The Nelder-Mead Simplex Method

With a few changes in the notation, this subsection is taken from (Rao, 1996). This method, first proposed by Spendley et al. (1962), and later improved by Nelder and Mead (1965), is based on the concept of *simplex* $\mathcal{S}_n$. A simplex is a $(n+1)$-vertex hyperpolyhedron in $\mathbb{R}^n$. The search for the minimum of the objective function $f(\mathbf{x})$ is conducted by means of function evaluations at all $n+1$ vertices of the simplex. The strategy followed is outlined below.

We start by defining an *initial simplex*, which is done by means of a *base point* $P_0 \in \mathbb{R}^n$, of position vector $\mathbf{x}_0$. The remaining n vertices of $\mathcal{S}_n$ are generated so as to yield a *regular hyperpolyhedron* of unit-length edges. To this end, let

$$p = \frac{1}{\sqrt{2}\,n}(\sqrt{n+1} + n - 1), \quad q = \frac{1}{\sqrt{2}\,n}(\sqrt{n+1} - 1) \tag{4.11a}$$

Then, if $\mathbf{e}_i$ denotes the unit vector in the direction of the ith coordinate axis, corresponding to x_i, let

$$\mathbf{x}_i = \mathbf{x}_0 + p\mathbf{e}_i + \sum_{j=1, j\neq i}^{n} q\mathbf{e}_j, \quad i = 1, 2, \ldots, n \tag{4.11b}$$

Shown in Fig. 4.1 are displays of the simplexes in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively, defined as described in eqs.(4.11a & 4.11b), with the base point *at the origin*.

The search strategy is based on three operations: a) *reflection*; (b) *contraction*; and (c) *expansion*.

Reflection

Let $f_i = f(\mathbf{x}_i)$, for $i = 0, 1, \ldots, n$, and

$$f_M = \max_i\{\, f_i \,\}_0^n, \quad f_m = \min_i\{\, f_i \,\}_0^n \tag{4.12}$$

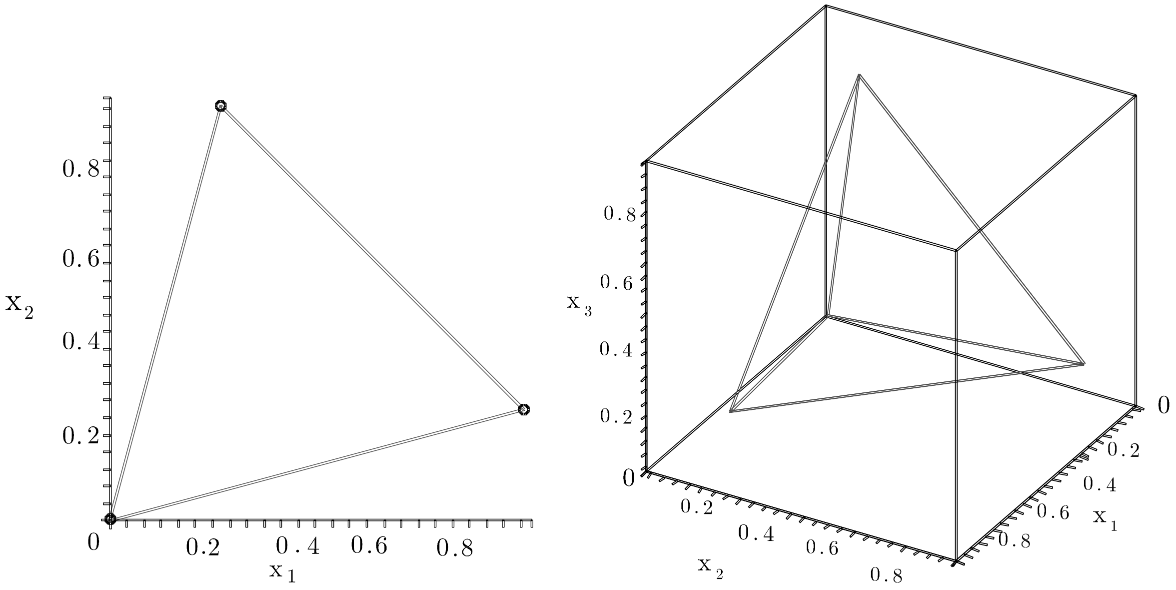


Figure 4.1: The initial simplex in $\mathbb{R}^2$ The initial simplex in $\mathbb{R}^3$

the corresponding vertices being P_M and P_m, of position vectors $\mathbf{x}_M$ and $\mathbf{x}_m$, respectively. With the foregoing information, we now seek a new simplex, by replacing *the worst* vertex P_M of the current simplex by a new one, P_{n+1}, of position vector $\mathbf{x}_{n+1}$. The new vertex is found by means of a reflection of P_M about the centroid $\overline{P}$, of position vector $\overline{\mathbf{x}}$, of all the simplex vertices, except for P_M, namely,

$$\overline{\mathbf{x}} = \frac{1}{n} \sum_{i=0,\, i\neq M}^{n} \mathbf{x}_i \tag{4.13}$$

Let, moreover, $\alpha > 0$ be the user-prescribed *reflection coefficient*, which is used to define the new vertex a distance $\alpha\|\overline{\mathbf{x}} - \mathbf{x}_M\|$ from the centroid $\overline{P}$, the position vector $\mathbf{x}_{n+1}$ of the new vertex thus being

$$\mathbf{x}_{n+1} = \overline{\mathbf{x}} + \alpha(\overline{\mathbf{x}} - \mathbf{x}_M) \tag{4.14}$$

The effect will be that P_{n+1} will lie *the farthest* from the worst vertex P_M, and hence, the new vertex is *very likely* to be *the best* of all the vertices of the new simplex, $\{\, P_i \,\}_{i=0,\ i\neq M}^{n+1}$. An unlikely, although quite possible scenario, is that $f_{n+1} = f_M$, and hence, no gain will be made by defining the new simplex. In this case, we can define the new vertex in one of two possible ways:

1. P_M is preserved, the rejected vertex being $P_{M'}$, which is the next worst vertex, i.e., with the subscript M' defined such that

$$f_{M'} = \max_i \{\, f_i \,\}_{i=0,\ i\neq M}^{n} \tag{4.15}$$

2. Alternatively, redefine α—make it either larger or smaller—while rejecting always the same worst vertex P_M.

Expansion

If the outcome of the reflection stage yields $f_{n+1} \equiv f(\mathbf{x}_{n+1}) < f_m$, then the direction defined by the vector difference $\mathbf{x}_{n+1} - \overline{\mathbf{x}}$ is very likely to point towards the minimum, and hence, it may be advisable to place a new vertex P_e away from $\overline{P}$ in the said direction, i.e., by defining the position vector $\mathbf{x}_e$ of the new vertex P_e in the form

$$\mathbf{x}_e = \overline{\mathbf{x}} + \gamma(\mathbf{x}_{n+1} - \overline{\mathbf{x}}) \tag{4.16}$$

$\gamma > 1$, for expansion.

Now, there are two possible outcomes:

- If $f_e \equiv f(\mathbf{x}_e) < f(\mathbf{x}_m)$, then replace P_M by P_e and start a new reflection;
- if $f_e > f(\mathbf{x}_m)$, then the expansion failed, and the new simplex obtained by the reflection, with P_M replaced by P_{n+1}, is kept, and a new reflection is started.

The alternative outcome of the reflection is described below.

Contraction

If $f_{n+1} > f_i \equiv f(\mathbf{x}_i)$, for $i = 0, 1, \ldots, M-1, M+1, \ldots, n$, and $f_{n+1} < f_M$, then we replace P_M by P_{n+1}. In this case, we define a new vertex P_c, of position vector $\mathbf{x}_c$, a distance $\beta\|\mathbf{x}_M - \overline{\mathbf{x}}\|$ from $\overline{P}$, with $0 \leq \beta \leq 1$. That is,

$$\mathbf{x}_c = \overline{\mathbf{x}} + \beta(\mathbf{x}_M - \overline{\mathbf{x}}) \tag{4.17}$$

which can be readily proven to yield a P_c lying between P_M and $\overline{P}$, for $\mathbf{x}_c$ has been defined as a *convex combination*[2] of $\mathbf{x}_c$ and $\overline{\mathbf{x}}$. Now,

- If $f_c \equiv f(\mathbf{x}_c) > f_M$, then no vertex of the simplex is changed, and we keep the new vertex with P_{n+1}, as obtained in the reflection operation;
- if $f_c < \min\{f_M, f_{n+1}\}$, then P_M is replaced by P_c, a new simplex thus being obtained, and a new reflection operation is started;

[2]See Section 5.4 for a definition of this term.

- if $f_c > \min\{f_M,\, f_{n+1}\}$, then the contraction failed, in which case P_i is replaced by $\overline{P}_i$, of position vector $\overline{\mathbf{x}}_i$, halfway between P_i and P_m, i.e.,

$$\overline{\mathbf{x}}_i = \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_m) \tag{4.18}$$

thereby defining a new simplex, and a new reflection is started.

Convergence criterion

The method converges when the rms value f_{rms} of the objective function is smaller than a prescribed tolerance ϵ, i.e., when

$$\sqrt{\frac{1}{n+1}\sum_{1}^{n+1}|f_i - \overline{f}|^2} \ < \ \epsilon \tag{4.19}$$

where (i) a relabelling of the vertices has been assumed, with the order $f_1 \leq f_2 \leq \ldots \leq f_{n+1}$ and (ii) $\overline{f}$ is the mean value of the objective function evaluated at all the vertices of the current complex, i.e.,

$$\overline{f} \equiv \frac{1}{n+1}\sum_{1}^{n+1} f_i \tag{4.20}$$

4.5 Gradient Methods

4.5.1 The Method of Steepest Descent (Cauchy)

Algorithm:

1. `Pick up an` *`initial guess`* x^1 `to start the iterations. Set the iteration counter` i `at` $i = 1$

2. `Define the` i`th` *`search direction`* s_i `as`

$$\mathbf{s}^i = -\nabla f\,|_{\mathbf{x}=\mathbf{x}^i} \tag{4.21}$$

3. `Define the next test point,` $\mathbf{x}^{i+1}$`, as`

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \lambda\mathbf{s}^i = \mathbf{x}^i - \lambda\nabla f\,\big|_{\mathbf{x}} = \mathbf{x}^i \tag{4.22}$$

`To find` λ`, conduct a` *`one-dimensional search`* `along the direction` s^i `so that` λ_{opt} `is the value of` λ `that minimizes` $F(\lambda) = f(\mathbf{x}^i - \lambda\nabla f\,|_{\mathbf{x}=\mathbf{x}^i})$

4. If $\mathbf{x}^{i+1}$ satisfies the *convergence criteria* adopted at the outset, **stop**; else, go to step 5.

5. Update the iteration counter: $i+1 \quad \leftarrow \quad i$. Go to step 2.

Convergence criteria: Use one or more of those applicable, namely,

$$|f(\mathbf{x}^{i+1}) - f(\mathbf{x}^i)| \leq \epsilon_1 |f(\mathbf{x}^i)| \tag{4.23a}$$

$$\|\nabla f\,|_{\mathbf{x}=\mathbf{x}^i}\,\| \leq \epsilon_2 \tag{4.23b}$$

$$\|\mathbf{x}^{i+1} - \mathbf{x}^i\| \leq \epsilon_3 \tag{4.23c}$$

Remark: In criteria (4.23b & c), *any* norm can be used $\Rightarrow$ Use the most economic one, i.e., the Chebyshev or maximum norm.

4.5.2 The Conjugate-Gradient Method (Fletcher-Reeves)

We use here the concept of $\mathbf{A}$-conjugacy introduced in Subsection 4.4.2, for a symmetric, positive-definite $\mathbf{A}$.

Preliminary Remarks:

- The conjugate-gradient method of Fletcher-Reeves (FR method) is aimed at minimizing a C^2-continuous function $f(\mathbf{x})$ under no constraints.

- The FR method works on the concept of *sequential quadratic programming* (SQP).

- The FR method is based on the *quadratic approximation* of $f(\mathbf{x})$: It is assumed that

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + f_0 + \text{HOT} \tag{4.24}$$

  where HOT stands for *higher-order terms*

- It is assumed that $\mathbf{A}$ in eq.(4.24) is *positive-definite*

Algorithm Overview: The FR algorithm works on the base of two items:

- A search direction $\mathbf{s}_i$ at each iteration, and

- a step of optimum length λ^* in direction $\mathbf{s}_i$ that minimizes f *in that direction*

Begin with an initial guess $\mathbf{x}^1$

Next, a *search direction* $\mathbf{s}_1$ is defined in the direction of *steepest descent* of $f(\mathbf{x})$, i.e., along $-\nabla f$ at $\mathbf{x}^1$, namely,

$$\mathbf{s}^1 \equiv -\nabla f\,|_{\mathbf{x}=\mathbf{x}^1} \tag{4.25}$$

Further, a new iterate, $\mathbf{x}^2$ is sought along the above direction, from the current iterate $\mathbf{x}^1$:

$$\mathbf{x}^2 \equiv \mathbf{x}^1 + \lambda \mathbf{s}^1 \tag{4.26}$$

where λ is a real number, as yet to be determined. This is done by imposing that ∇f, when evaluated at $\mathbf{x} = \mathbf{x}^2$, be *normal* to $\mathbf{s}^1$, i.e.,

$$(\nabla f\,\big|_{\mathbf{x}=\mathbf{x}^2})^T\,\mathbf{s}^1 = 0 \tag{4.27}$$

Now susbtitute $\mathbf{x}^2$ as given by eq.(4.26) and the quadratic approximation of $f(\mathbf{x})$ into eq.(4.27), to obtain, with $\nabla f \approx \mathbf{A}\mathbf{x} + \mathbf{b}$,

$$[\mathbf{A}(\underbrace{\mathbf{x}^1 + \lambda \mathbf{s}^1}_{\mathbf{x}^2}) + \mathbf{b}]^T \mathbf{s}^1 = 0 \tag{4.28}$$

whence the optimum value of λ, λ^*, is readily derived:

$$\lambda^* = \frac{-(\mathbf{A}\mathbf{x}^1 + \mathbf{b})^T \mathbf{s}^1}{(\mathbf{s}^1)^T \mathbf{A} \mathbf{s}^1} \equiv -\frac{(\mathbf{s}^1)^T \nabla f\,|_{\mathbf{x}=\mathbf{x}^1}}{(\mathbf{s}^1)^T \mathbf{A} \mathbf{s}^1} \tag{4.29}$$

where, from the assumed positive-definiteness of $\mathbf{A}$,

$$(\mathbf{s}^1)^T \mathbf{A} \mathbf{s}^1 > 0$$

and hence, $\mathbf{s}^1$ can be expressed, upon recalling eq.(4.26), as

$$\mathbf{s}^1 = \frac{1}{\lambda^*}(\mathbf{x}^2 - \mathbf{x}^1) \tag{4.30}$$

The new search direction, $\mathbf{s}^2$, is defined as a *linear combination* of $\mathbf{s}^1$ and $-\nabla f\,|_{\mathbf{x}=\mathbf{x}^2}$, i.e.,

$$\mathbf{s}^2 \equiv -\nabla f\,\big|_{\mathbf{x}=\mathbf{x}^2} + \beta_2 \mathbf{s}^1 \tag{4.31}$$

where β_2 is chosen so as to make $\mathbf{s}^2$ conjugate to $\mathbf{s}^1$ with respect to $\mathbf{A}$:

$$(\mathbf{s}^1)^T \mathbf{A} \mathbf{s}^2 = 0 \quad \Rightarrow \quad (\mathbf{s}^1)^T \mathbf{A}(-\nabla f\,|_{\mathbf{x}=\mathbf{x}^2} + \beta_2 \mathbf{s}^1) = 0$$

or

$$(-\nabla f\,|_{\mathbf{x}=\mathbf{x}^2} + \beta_2 \mathbf{s}^1)^T \mathbf{A} \mathbf{s}^1 = 0 \tag{4.32}$$

Recall eq.(4.30) and substitute that expression into eq.(4.32), to obtain

$$(-\nabla f\,|_{\mathbf{x}=\mathbf{x}^2} + \beta_2 \mathbf{s}^1)^T \mathbf{A} \left[\frac{1}{\lambda^*}(\mathbf{x}^2 - \mathbf{x}^1)\right] = 0 \tag{4.33}$$

Next, we find an expression for the difference $\mathbf{x}^2 - \mathbf{x}^1$ in terms of the gradients at $\mathbf{x}^1$ and $\mathbf{x}^2$. Indeed, recalling the quadratic approximation, eq.(4.24),

$$\nabla f\,|_{\mathbf{x}=\mathbf{x}^2} - \nabla f\,|_{\mathbf{x}=\mathbf{x}^1} \approx \mathbf{A}\mathbf{x}^2 - \mathbf{b} - (\mathbf{A}\mathbf{x}^1 - \mathbf{b}) = \mathbf{A}(\mathbf{x}^2 - \mathbf{x}^1) \tag{4.34}$$

Substitute expression (4.34) into eq.(4.33), after clearing the denominator:

$$(-\nabla f\,|_{\mathbf{x}=\mathbf{x}^2} + \beta_2 \mathbf{s}^1)^T (\nabla f\,|_{\mathbf{x}=\mathbf{x}^2} - \nabla f\,|_{\mathbf{x}=\mathbf{x}^1}) = 0 \tag{4.35}$$

Upon expansion,

$$-(\nabla f\,|_{\mathbf{x}=\mathbf{x}^2})^T \nabla f\,|_{\mathbf{x}=\mathbf{x}^2} + \underbrace{(\underbrace{\nabla f\,|_{\mathbf{x}=\mathbf{x}^1}}_{-\mathbf{s}^1})^T \nabla f\,|_{\mathbf{x}=\mathbf{x}^2}}_{0} + \beta_2 \underbrace{(\nabla f\,|_{\mathbf{x}=\mathbf{x}^2})^T \mathbf{s}^1}_{0} - \beta_2 (\nabla f\,|_{\mathbf{x}=\mathbf{x}^1})^T \mathbf{s}^1 = 0$$

where we have recalled eqs.(4.25) and (4.27). Hence, the above equation simplifies to

$$(\nabla f\,|_{\mathbf{x}=\mathbf{x}^2})^T \nabla f\,|_{\mathbf{x}=\mathbf{x}^2} + \beta_2 (\nabla f\,|_{\mathbf{x}=\mathbf{x}^1})^T \underbrace{\mathbf{s}^1}_{-\nabla f|_{\mathbf{x}=\mathbf{x}^2}} = 0$$

whence we can solve for β_2 as

$$\beta_2 = \frac{(\nabla f\,|_{\mathbf{x}=\mathbf{x}^2})^T \nabla f\,|_{\mathbf{x}=\mathbf{x}^2}}{(\nabla f\,|_{\mathbf{x}=\mathbf{x}^1})^T \mathbf{s}^1} \equiv \frac{(\nabla f\,|_{\mathbf{x}=\mathbf{x}^2})^T \nabla f\,|_{\mathbf{x}=\mathbf{x}^2}}{(\nabla f\,|_{\mathbf{x}=\mathbf{x}^1})^T \nabla f\,|_{\mathbf{x}=\mathbf{x}^1}}$$

or

$$\beta_2 = \frac{\|\nabla f\,|_{\mathbf{x}=\mathbf{x}^2}\|^2}{\|\nabla f\,|_{\mathbf{x}=\mathbf{x}^1}\|^2} \tag{4.36}$$

A third search direction $\mathbf{s}^3$ is now defined as a linear combination of $\mathbf{s}^2$ and $-\nabla f\,|_{\mathbf{x}=\mathbf{x}^3}$:

$$\mathbf{s}^3 = -\nabla f\,|_{\mathbf{x}=\mathbf{x}^3} + \beta_3 \mathbf{s}^2 \tag{4.37}$$

Now impose the *conjugacy condition* (4.6):

$$(\mathbf{s}^2)^T \mathbf{A} \mathbf{s}^3 = 0 \quad \Rightarrow \quad \beta_3 = \frac{(\nabla f\,|_{\mathbf{x}=\mathbf{x}^3})^T \nabla f\,|_{\mathbf{x}=\mathbf{x}^3}}{(\nabla f\,|_{\mathbf{x}=\mathbf{x}^2})^t \nabla f\,|_{\mathbf{x}=\mathbf{x}^2}} = \frac{\|\nabla f\,|_{\mathbf{x}=\mathbf{x}^3}\|^2}{\|\nabla f\,|_{\mathbf{x}=\mathbf{x}^2}\|^2} \tag{4.38}$$

In general, we have

$$\mathbf{s}^i = -\nabla f\,|_{\mathbf{x}=\mathbf{x}^i} + \beta_i \mathbf{s}^i, \quad \beta_i = \frac{\|\nabla f\,|_{\mathbf{x}=\mathbf{x}^i}\|^2}{\|\nabla f\,|_{\mathbf{x}=\mathbf{x}^{i-1}}\|^2}, \quad i = 2, 3, , \ldots \tag{4.39}$$

The procedure stops when $\|\nabla f\,|_{\mathbf{x}=\mathbf{x}^i}\| < \epsilon$, for a user-prescribed tolerance ϵ, which indicates that the normality condition (4.2a) has been satisfied.

Summary of the Fletcher-Reeves Algorithm

1. Choose an initial guess $\mathbf{x}^1$

2. Let $\mathbf{s}^1 = -\nabla f\,|_{\mathbf{x}=\mathbf{x}^1}$

3. Let
$$\mathbf{x}^2 = \mathbf{x}^1 + \lambda^* \mathbf{s}^1 \tag{4.40}$$
where λ^* is the value of λ that makes $(\mathbf{s}^1)^T \nabla f\,|_{\mathbf{x}=\mathbf{x}^2} = 0$

4. Let $i = 2$

5. Let
$$\mathbf{s}^i = -\nabla f\,|_{\mathbf{x}=\mathbf{x}^i} + \frac{\|\nabla f\,|_{\mathbf{x}=\mathbf{x}^i}\|^2}{\|\nabla f\,|_{\mathbf{x}=\mathbf{x}^{i-1}}\|^2}\mathbf{s}^{i-1}$$

6. Find the value of λ, λ^*, that makes $(\mathbf{s}^i)^T \nabla f\,|_{\mathbf{x}=\mathbf{x}^{i+1}} = 0$. Then,
$$\mathbf{x}^{i+1} = \mathbf{x}^i + \lambda^* \mathbf{s}^i$$

7. If $\|\nabla f\,|_{\mathbf{x}=\mathbf{x}^{i+1}}\| < \epsilon$, stop; else $i \leftarrow i+1$ and go to step 5

4.5.3 Quasi-Newton Methods

As we will see in Section 4.6, Newton methods rely on the normality conditions, which lead to a determined system of n nonlinear equations in n unknowns. In applying those methods it is assumed that the Hessian of the objective function, which is the Jacobian $\mathbf{\Phi}$ of the Newton-Raphson method, is available, and hence, the Hessian can be used to update the iterations. Quasi-Newton methods replace the update $\Delta\mathbf{x} = -\mathbf{\Phi}_0^{-1}\boldsymbol{\phi}^0$ of eq.(3.80) by an expression that a) does not rely on the Hessian, but only on the gradient of the objective function and b) does not require any matrix inversion. These features make quasi-Newton methods quite attractive, and many times, preferable over Newton methods. The two quasi-Newton methods outlined below differ only in the form in which the update of the solution is computed. These two methods aim at finding an approximation to $(\nabla\nabla f)^{-1}$ using only information on ∇f.

Moreover, while the Newton-Raphson method is known to have a *quadratic convergence rate*, quasi-Newton methods show a convergence rate that lies between gradient methods and Newton methods. That is, quasi-Newton methods have a *superlinear convergence rate*.

The Davidon-Fletcher-Powell Method

The method is summarized below:

Algorithm

1. `Give an initial guess:` $\mathbf{x}^0$
2. `Define an initial search direction:` $\mathbf{s}^0 = -\nabla f|_{\mathbf{x}=\mathbf{x}^0}$
3. `Define an initial Hessian-inverse:` $\mathbf{B}_0 = \mathbf{1}$, `the` $n \times n$ `identity matrix`
4. `Find` λ_i `that minimizes`[3] $f(\mathbf{x}^{i+1})$. `Then, let`

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \lambda_i \mathbf{B}_i \nabla f|_{\mathbf{x}=\mathbf{x}^i}$$

5. $$\mathbf{g}^i = \nabla f|_{\mathbf{x}=\mathbf{x}^{i+1}} - \nabla f|_{\mathbf{x}=\mathbf{x}^i}$$

6. $$\mathbf{M}_i = \lambda_i \frac{\mathbf{s}^i(\mathbf{s}^i)^T}{(\mathbf{s}^i)^T\mathbf{g}^i}, \quad \mathbf{N}_i = -\frac{\mathbf{B}_i\mathbf{g}^i(\mathbf{B}_i\mathbf{g}^i)^T}{(\mathbf{g}^i)^T\mathbf{B}_i\mathbf{g}^i}, \quad \mathbf{B}_{i+1} = \mathbf{B}_i + \mathbf{M}_i + \mathbf{N}_i$$

7. $$\mathbf{s}^{i+1} = \mathbf{B}_{i+1}\mathbf{g}^i$$

8. `if convergence criterion reached, stop; else, go to 4`

The Broyden-Fletcher-Goldfarb-Shanno Method

This is an improved DFP method, but still with superlinear convergence. Only difference with the DFP Algorithm lies in step 6, which is replaced by:

$$\begin{aligned}
\mathbf{M}'_i &= \alpha_i \frac{\mathbf{s}^i(\mathbf{s}^i)^T}{(\mathbf{s}^i)^T\mathbf{g}^i}, \quad \alpha_i = 1 + \frac{(\mathbf{g}^i)^T\mathbf{B}_i\mathbf{g}^i}{(\mathbf{s}^i)^T\mathbf{g}^i} \\
\mathbf{N}'_i &= -\frac{\mathbf{s}^i(\mathbf{g}^i)^T\mathbf{B}_i}{(\mathbf{s}^i)^T\mathbf{g}^i} \\
\mathbf{B}_{i+1} &= \mathbf{B}_i + \mathbf{M}'_i + \mathbf{N}'_i + (\mathbf{N}'_i)^T
\end{aligned}$$

[3] In this step, any of the methods studied in Chapter 2 can be applied. A thorough discussion of univariable minimization is available in (Brent, 1972).

4.6 Newton Methods

4.6.1 The Newton-Raphson Method

Here, we resort to the normality condition (4.2a), and let

$$\boldsymbol{\phi}(\mathbf{x}) \equiv \nabla f$$

the normality condition thus leading to a system of n nonlinear equations in n unknowns of the form of eq.(3.75), repeated below for quicfk reference:

$$\boldsymbol{\phi}(\mathbf{x}) = \mathbf{0}$$

which can be solved using the Newton-Raphson method because, by assumption, second-order derivatives of the objective function are available, and hence, the Jacobian $\boldsymbol{\Phi}$ of $\boldsymbol{\phi}(\mathbf{x})$ with respect to $\mathbf{x}$ is nothing but the Hessian matrox of $f(\mathbf{x})$, i.e.,

$$\boldsymbol{\Phi} = \nabla\nabla f$$

This method, while offering a quadratic convergence, is not as favoured as methods of the gradient type, mostly because of the cost of solving a system of linear equations, namely, eq.(3.79) at each iteration.

4.6.2 The Levenberg-Marquardt Method

The Levenberg-Marquardt method aims at enhancing the robustness of the Newton-Raphson method, when the Hessian becomes ill-conditioned, by adding to the Hessian, which is assumed positive-definite, a symmetric, isotropic matrix $\alpha\mathbf{1}$, where $\alpha > 0$ and $\mathbf{1}$ is the $n \times n$ identity matrix:

$$\overline{\nabla\nabla f} \quad \leftarrow \quad \nabla\nabla f + \alpha\mathbf{1} \tag{4.41}$$

Notice that the eigenvalues[4] of $\nabla\nabla f$, denoted by $\{\, \lambda_i \,\}_1^n$, and those of $\overline{\nabla\nabla f}$, denoted by $\{\, \mu_i \,\}_1^n$, are related by

$$\mu_i = \lambda_i + \alpha, \quad i = 1, 2, \ldots, n$$

If we denote by κ the 2-norm condition number of $\nabla\nabla f$ and by $\overline{\kappa}$ that of $\overline{\nabla\nabla f}$, we have

$$\kappa = \frac{\lambda_M}{\lambda_m}, \quad \overline{\kappa} = \frac{\lambda_M + \alpha}{\lambda_m + \alpha} \tag{4.42}$$

[4]By virtue of the assumed positive-definiteness of the Hessian, its eigenvalues are identical to their singular values.

the result being that $\overline{\kappa} < \kappa$, and hence, the numerical behaviour of the Hessian is stabilized.

Chapter 5

Equality-Constrained Optimization: Normality Conditions

5.1 Introduction

In this chapter we solve the simplest class of constrained-optimization problems, namely, those subject to equality constraints. The problem statement at hand is

$$f(\mathbf{x}) \quad \rightarrow \quad \min_{\mathbf{x}} \tag{5.1a}$$

subject to

$$\mathbf{h}(\mathbf{x}) = \mathbf{0}_l \tag{5.1b}$$

where $\mathbf{h}$ is a *smooth*[1] l-dimensional vector function of the n-dimensional vector argument $\mathbf{x}$, $\mathbf{0}_l$ denoting the l-dimensional zero vector. Moreover, $l < n$, for an n-dimensional design vector $\mathbf{x}$.

The main outcome is the derivation of the *normality conditions* of the problems at hand. We derive these in two forms: (i) the *direct form*, in terms of the gradients of the objective function $f(\mathbf{x})$ to be minimized and of the constraints (5.1b); and (ii) the *dual form*, in terms of an *orthogonal complement* of the gradient of $\mathbf{h}$ with respect to $\mathbf{x}$.

As a special case, that lends itself to a closed-form solution, we study *minimum-norm* problems, whereby a *weighted* Euclidean norm of the design vector is to be

[1]Smoothness implies that $\mathbf{h}(\mathbf{x})$ is continuous and has a continuous gradient with respect to $\mathbf{x}$.

minimized subject to l linear equality constraints. In this vein, we introduce the *right Moore-Penrose generalized inverse.*

5.2 The First-Order Normality Conditions

5.2.1 The Primal Form

We now derive the normality conditions of problem (5.1a) in *primal form.* To this end, we resort to *Lagrange multipliers* $\lambda_1, \lambda_2, \ldots, \lambda_l$, one for each scalar constraint $h_i(\mathbf{x}) = 0$, and group them in the l-dimensional array $\boldsymbol{\lambda}$. Upon adjoining the l constraints to the objective function $f(\mathbf{x})$, we obtain the *Lagrangian* $F(\mathbf{x}; \boldsymbol{\lambda})$ that we aim at minimizing under no constraints, while choosing $\boldsymbol{\lambda}$ in such a way that the l equality constraints are satisfied. That is,

$$F(\mathbf{x}; \boldsymbol{\lambda}) \equiv f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{h} \quad \rightarrow \quad \min_{\mathbf{x},\, \boldsymbol{\lambda}} \tag{5.2}$$

subject to no constraints. We thus have transformed the equality-constrained minimization problem into an unconstrained one. We derive now the normality conditions of the problem at hand by recalling those of Chapter 2, requiring that a) the gradient of the objective function with respect to the design variables vanish and b) the Hessian of the objective function with respect to the same variables be positive-definite. However, note that we now have l additional variables besides the original n design variables. We thus have to augment the design-variable vector correspondingly, which we do by defining an augmented $(n + l)$-dimensional design vector $\mathbf{y}$:

$$\mathbf{y} \equiv \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix} \tag{5.3}$$

Therefore, the unconstrained minimization problem (5.2) can be formulated in a more compact form, namely,

$$F(\mathbf{y}) \quad \rightarrow \quad \min_{\mathbf{y}} \tag{5.4}$$

subject to no constraints. The first-order normality conditions of the above problem are, thus,

$$\frac{\partial F}{\partial \mathbf{y}} = \begin{bmatrix} \partial F / \partial \mathbf{x} \\ \partial F / \partial \boldsymbol{\lambda} \end{bmatrix} = \mathbf{0}_{n+l} \tag{5.5a}$$

where $\mathbf{0}_{n+l}$ denotes the $(n+l)$-dimensional zero vector. Upon expansion, the above equation yields

$$\frac{\partial F}{\partial \mathbf{x}} = \mathbf{0}_n \tag{5.5b}$$

$$\frac{\partial F}{\partial \boldsymbol{\lambda}} = \mathbf{0}_l \tag{5.5c}$$

To gain insight into the geometric significance of the foregoing normality conditions, we expand the left-hand side of eq.(5.5b) componentwise:

$$\begin{aligned}
\frac{\partial F}{\partial x_1} &\equiv \frac{\partial f}{\partial x_1} + \lambda_1 \frac{\partial h_1}{\partial x_1} + \lambda_2 \frac{\partial h_2}{\partial x_1} + \cdots + \lambda_l \frac{\partial h_l}{\partial x_1} = 0 \\
\frac{\partial F}{\partial x_2} &\equiv \frac{\partial f}{\partial x_2} + \lambda_1 \frac{\partial h_1}{\partial x_2} + \lambda_2 \frac{\partial h_2}{\partial x_2} + \cdots + \lambda_l \frac{\partial h_l}{\partial x_2} = 0 \\
&\vdots \\
\frac{\partial F}{\partial x_n} &\equiv \underbrace{\frac{\partial f}{\partial x_n}}_{\nabla f} + \lambda_1 \frac{\partial h_1}{\partial x_n} + \lambda_2 \frac{\partial h_2}{\partial x_n} + \cdots + \lambda_l \frac{\partial h_l}{\partial x_n} = 0
\end{aligned} \tag{5.6}$$

where the the first term of the ith equation can be readily identified as the ith component of $\nabla f = \partial f/\partial \mathbf{x}$. The remaining terms of the same equation can be identified as the ith component of an inner product p_i defined as

$$p_i \equiv [\, \partial h_1/\partial x_i \quad \partial h_2/\partial x_i \quad \cdots \quad \partial h_l/\partial x_i \,] \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_l \end{bmatrix}$$

Therefore, if we let $\mathbf{p} = \begin{bmatrix} p_1 & p_2 & \dots & p_n \end{bmatrix}^T$, then

$$\mathbf{p} = \underbrace{\begin{bmatrix} \partial h_1/\partial x_1 & \partial h_2/\partial x_1 & \cdots & \partial h_l/\partial x_1 \\ \partial h_1/\partial x_2 & \partial h_2/\partial x_2 & \cdots & \partial h_l/\partial x_2 \\ \vdots & \vdots & \ddots & \vdots \\ \partial h_1/\partial x_n & \partial h_2/\partial x_n & \cdots & \partial h_l/\partial x_n \end{bmatrix}}_{(\nabla \mathbf{h})^T:\ n\times l} \underbrace{\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_l \end{bmatrix}}_{\boldsymbol{\lambda}} \tag{5.7}$$

which can be readily identified, in turn, as the product $(\nabla \mathbf{h})^T \boldsymbol{\lambda}$. The first n normality conditions, displayed in eq.(5.5b), thus amount to

$$\nabla f + \mathbf{J}^T \boldsymbol{\lambda} = \mathbf{0}_n \tag{5.8a}$$

where $\mathbf{J} \equiv \nabla \mathbf{h}$ is the gradient of $\mathbf{h}$ with respect to $\mathbf{x}$. The remaining l normality conditions, displayed in eq.(5.5c), yield nothing but the constraints themselves, namely

$$\mathbf{h}(\mathbf{x}) = \mathbf{0}_l \tag{5.8b}$$

Equation (5.8a) is the vector representation of the *first-order normality conditions* (FONC) sought. What eq.(5.8a) represents has a geometric significance that will be made apparent upon rewriting it in the alternative form

$$\mathbf{J}^T \boldsymbol{\lambda} = -\nabla f \tag{5.9}$$

The foregoing equation states that, at a stationary point $\mathbf{x}_0$, $-\nabla f$, or ∇f for that matter, lies in the range of the transpose of the gradient of the constraints. Notice that the range $\mathcal{J}'$ of $\mathbf{J}^T$ is a subspace of the n-dimensional space of design variables. In fact, $\dim[\mathcal{R}(\mathbf{J}^T)] = l < n$, for this subspace is spanned by l linearly independent vectors, the columns of $\mathbf{J}^T$, or the n-dimensional rows of $\mathbf{J}$.

Algebraically, what eq.(5.9) represents is an overdetermined system of n linear equations in the $l < n$ unknowns $\{ \lambda_i \}_1^l$. The normality condition then states that the least-square approximation of this overdetermined system yields a zero error. That is, at a stationary point, the n $(> l)$ equations (5.9) become all consistent. Note that the least-square approximation $\boldsymbol{\lambda}_0$ of the foregoing equations can be expressed in terms of the left Moore-Penrose generalized inverse of $\mathbf{J}^T$, namely,

$$\boldsymbol{\lambda}_0 = -(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{J}\nabla f \tag{5.10}$$

The least-square error $\mathbf{e}_0$ of this approximation is thus

$$\mathbf{e}_0 = \mathbf{J}^T\boldsymbol{\lambda}_0 - (-\nabla f) = -\mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{J}\nabla f + \nabla f = [\mathbf{1} - \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{J}]\nabla f \tag{5.11}$$

with $\mathbf{1}$ denoting the $n \times n$ identity matrix.

We can now express the first-order normality condition (5.9) in yet one more alternative form:

$$[\mathbf{1} - \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{J}]\nabla f = \mathbf{0}_n \tag{5.12}$$

The matrix inside the brackets in the foregoing equation can be readily identified as a projector. This projector maps vectors in $\mathbb{R}^n$ onto the nullspace of $\mathbf{J}$. In other words, *at a stationary point* P_0 the gradient of the objective function need not vanish; only its projection onto the nullspace of the gradient of the constraints must vanish, which is an alternative form of stating the first-order normality condition. Sometimes the product $\overline{\nabla} f$, defined as

$$\overline{\nabla} f \equiv [\mathbf{1} - \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{J}]\nabla f \, \in \mathbb{R}^n \tag{5.13}$$

is referred to as the *constrained gradient.* The FONC (5.12) can then be simply stated as: *At a stationary of the equality-constrained problem (5.1a & b), the constrained gradient vanishes.*

Exercise 5.2.1
Prove that

$$\mathbf{P} \equiv \mathbf{1} - \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{J} \tag{5.14}$$

indeed maps vectors in $\mathbb{R}^n$ onto the nullspace of $\mathbf{J}$ and that $\mathbf{P}$ is a projector.

5.2.2 The Dual Form

One more form of the FONC of the problem at hand is now derived in what we can term *dual form.* To this end, we realize that the solution sought $\mathbf{x}_0$ must lie in a subset of the space $\mathbb{R}^n$ of design variables, of reduced dimension $n-l$, which contains all design vectors $\mathbf{x}$ satisfying the constraints. Now, this set need not be a vector space, and in general it is not. Indeed, if the constraints are nonlinear, then the sum of two distinct vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ will most likely not satisfy the constraints, even if these two designs do so independently. Neither need the zero vector satisfy the constraints, which thus disqualifies the set from being a subspace of $\mathbb{R}^n$. What we have as a *feasible subset* of the design space is a *manifold* $\mathcal{F}$, i.e., a smooth surface embedded in $\mathbb{R}^n$. We shall term this subset the *feasible manifold.*

Finding $\mathcal{F}$ may be a tremendous task when the constraints are nonlinear and algebraically complicated. The good news is that we do not actually need the feasible manifold to obtain a feasible solution. What we really need is a *feasible subspace* tangent to the said manifold at a feasible point. We discuss below how to obtain this subspace. Assume that we have a feasible point P_F, of position vector $\mathbf{x}_F$, i.e.,

$$\mathbf{h}(\mathbf{x}_F) = \mathbf{0}_l \tag{5.15}$$

An arbitrary "move" $\Delta\mathbf{x}$ from $\mathbf{x}_F$ will most likely take P_F away from the constraints $\mathbf{h}(\mathbf{x}) = \mathbf{0}_l$. What we would like to find is a *feasible move*, i.e., a vector $\Delta\mathbf{x}_F$ such that $\mathbf{x}_F + \Delta\mathbf{x}_F$ still verifies the constraints, i.e.,

$$\mathbf{h}(\mathbf{x}_F + \Delta\mathbf{x}_F) = \mathbf{h}(\mathbf{x}_F) + \mathbf{J}(\mathbf{x}_F)\Delta\mathbf{x}_F + \mathrm{HOT} = \mathbf{0}_l \tag{5.16}$$

Since we assumed at the outset that $\mathbf{x}_F$ is feasible, eq.(5.15), we have, from the foregoing equation and to a first-order approximation, i.e., neglecting HOT,

$$\mathbf{J}(\mathbf{x}_F)\Delta\mathbf{x}_F = \mathbf{0}_l \tag{5.17}$$

Moreover, $\mathbf{J}$ is of $l \times n$, with $l < n$, and hence, it is possible to find $n - l$ linearly independent vectors of $\mathbb{R}^n$ lying in $\mathcal{N}(\mathbf{J}(\mathbf{x}_F))$, i.e., in the nullspace of $\mathbf{J}(\mathbf{x}_F)$. These vectors can be produced in many ways. We will not dwell, for the time being, in the pertinent computing methods, but rather on the concepts behind the production of those $n - l$ vectors. Let us thus assume that we have found such $n - l$ linearly independent vectors, arrayed in the $n \times n'$ matrix $\mathbf{L}$, with $n' \equiv n - l$, and hence,

$$\mathbf{J}\mathbf{L} = \mathbf{O}_{ln'} \tag{5.18}$$

matrix $\mathbf{L}$ being termed an *orthogonal complement* of $\mathbf{J}$ and $\mathbf{O}_{ln'}$ denoting the $l \times n'$ zero matrix.

Now, if we define

$$\Delta\mathbf{x}_F = \mathbf{L}\Delta\mathbf{u} \tag{5.19a}$$

for arbitrary $\Delta\mathbf{u} \in \mathbb{R}^{n'}$, we will have

$$\mathbf{J}\Delta\mathbf{x}_F = \mathbf{J}\mathbf{L}\Delta\mathbf{u} = \mathbf{0}_{n'} \tag{5.19b}$$

the "move" $\Delta\mathbf{x}_F$ thus verifying the constraints to a first degree. Now, the first-order normality condition of the problem at hand can be cast in the form

$$\Delta f \equiv (\nabla f)^T \Delta\mathbf{x}_F = (\nabla f)^T \mathbf{L}\Delta\mathbf{u} = (\mathbf{L}^T \nabla f)^T \Delta\mathbf{u} = 0 \quad \forall \quad \Delta\mathbf{u}$$

Hence, the alternative form of the FONC is

$$\mathbf{L}^T \nabla f = \mathbf{0}_{n'} \tag{5.20}$$

That is, at a stationary point, the gradient of f need not vanish; it must lie in the nullspace of $\mathbf{L}^T$, i.e., in the range of $\mathbf{L}$. We can thus call $\mathbf{L}^T \nabla f$ the *feasible gradient*, and represent it by $\nabla_u f$, i.e.,

$$\nabla_u f = \mathbf{L}^T \nabla f \tag{5.21}$$

which is a $(n - l)$-dimensional vector. Notice that, from eq.(5.19a), $\mathbf{L}$ has the differential interpretation

$$\mathbf{L} = \frac{\partial \mathbf{x}}{\partial \mathbf{u}} \tag{5.22}$$

and hence, the FONC (5.20) can be restated as

$$\left(\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\right)^T \left(\frac{\partial f}{\partial \mathbf{x}}\right) \equiv \frac{\partial f}{\partial \mathbf{u}} = \mathbf{0}_{n'} \quad \text{or} \quad \nabla_u f = \mathbf{0}_{n'} \tag{5.23}$$

That is, the FONC (5.23) states that, at a stationary point of problem (5.1a & b), the gradient of $f(\mathbf{x})$ with respect to the vector of independent design variables $\mathbf{u}$ vanishes.

Remark: When comparing the two forms of the FONC, eqs.(5.12) and (5.20), the simplicity of the latter with respect to the former is apparent. This simplicity, however, is more than formal, for eq.(5.12) involves n scalar equations, while eq.(5.20) involves only $n - l$ scalar equations.

5.3 The Second-Order Normality Conditions

5.3.1 The Primal Form

The second-order normality conditions (SONC) of the problem at hand require that the Hessian of the Lagrangian $F(\mathbf{x}; \boldsymbol{\lambda})$ with respect to the $(n + l)$-dimensional *augmented design-variable vector* $\boldsymbol{\xi} \equiv [\,\mathbf{x}^T \ \boldsymbol{\lambda}^T\,]^T$ be positive-definite. The said Hessian takes the form

$$\frac{\partial^2 F}{\partial \boldsymbol{\xi}^2} \equiv \mathbf{F} = \begin{bmatrix} \nabla\nabla f + \partial(\mathbf{J}^T\boldsymbol{\lambda})/\partial\mathbf{x} & \mathbf{J}^T \\ \mathbf{J} & \mathbf{O}_l \end{bmatrix} \tag{5.24}$$

where, as usual, $\nabla\nabla f$ represents the Hessian of $f(\mathbf{x})$ with respect to $\mathbf{x}$, and $\mathbf{O}_l$ denotes the $l \times l$ zero matrix. Notice that computing the above Hessian requires the computation of $\nabla\nabla f$ and $\partial(\mathbf{J}^T\boldsymbol{\lambda})/\partial\mathbf{x}$, which involves the second derivatives of $\mathbf{h}(\mathbf{x})$ with respect to $\mathbf{x}$. Moreover, in order to verify the second-order normality conditions, the $n + l$ eigenvalues of $\partial^2 F/\partial\boldsymbol{\xi}^2$ must be computed. We can now state the SONC in primal form: A stationary point $\mathbf{x}_0$

(i) is a local minimum if and only if $\mathbf{F}(\mathbf{x}_0)$ is positive-definite;

(ii) is a local maximum if and only if $\mathbf{F}(\mathbf{x}_0)$ is negative-definite;

(iii) is a saddle point if and only if $\mathbf{F}(\mathbf{x}_0)$ is sign-indefinite.

Thus, the second-order normality conditions in primal form are extremely costly to verify.

5.3.2 The Dual Form

Alternatively, we resort to the form (5.19a) of $\Delta\mathbf{x}_F$ and assume that we have found a stationary point verifying eq.(5.20). Upon expansion of $f(\mathbf{x}_F + \Delta\mathbf{x}_F)$ to second order, we obtain

$$\Delta f = f(\mathbf{x}_F + \Delta\mathbf{x}_F) - f(\mathbf{x}_F) \approx (\nabla f|_{x=\mathbf{x}_0})^T \Delta\mathbf{x}_F + \frac{1}{2}(\Delta\mathbf{x}_F)^T \nabla\nabla f\Big|_{x=\mathbf{x}_0} \Delta\mathbf{x}_F$$

which must be positive if the current value $\mathbf{x}_0$ is a minimum. Thus, upon substituting eq.(5.19a) into the above expression, we obtain

$$\Delta f \approx (\nabla f|_{x=\mathbf{x}_0})^T \mathbf{L}\Delta\mathbf{u} + \frac{1}{2}\Delta\mathbf{u}^T\mathbf{L}^T\nabla\nabla f\Big|_{x=\mathbf{x}_0} \mathbf{L}\Delta\mathbf{u} > 0$$

Now, since we have assumed that the FONC holds at the stationary point $\mathbf{x}_0$, eq.(5.20) holds, and hence,

$$(\nabla f|_{x=\mathbf{x}_0})^T\mathbf{L}\Delta\mathbf{u} = 0$$

the second-order normality condition thus becoming

$$\Delta\mathbf{u}^T\mathbf{L}^T\nabla\nabla f\big|_{x=\mathbf{x}_0}\mathbf{L}\Delta\mathbf{u} > 0 \quad \forall \quad \Delta\mathbf{u} \tag{5.25}$$

We term the product $\mathbf{L}^T\nabla\nabla f\big|_{x=\mathbf{x}_0}\mathbf{L}$ the *feasible Hessian* of f. That is, *a stationary point $\mathbf{x}_0$ is a local minimum if its feasible Hessian is positive-definite. As a consequence, then, at a minimum, the Hessian itself need not be positive-definite, but its feasible component must be.* This constitutes the dual form of the SONC.

We can now represent the $(n-l)\times(n-l)$ feasible Hessian as

$$\nabla_u\nabla_u f \equiv \mathbf{L}^T\nabla\nabla f\mathbf{L} = \left(\frac{\partial\mathbf{x}}{\partial\mathbf{u}}\right)^T(\nabla\nabla f)\frac{\partial\mathbf{x}}{\partial\mathbf{u}} \tag{5.26}$$

the SONC thus lending themselves to a more straightforward interpretation:

A stationary point $\mathbf{x}_0$ of problem (5.1a & b) is a minimum iff the Hessian of $f(\mathbf{x})$ with respect to the independent design-variable vector $\mathbf{u}$ is positive-definite.

Remark: At a minimum, the Hessian of f with respect to $\mathbf{x}$ need not be positive-definite. However, if $\nabla\nabla f$ is positive-definite, then $\nabla_u\nabla_u f$ is necessarily positive-definite as well.

Example 5.3.1 (The Design of a Positioning Robot for a Given Reach)
In designing the manipulator of Fig. 5.1 (Angeles, 2002), we want to find the value of the length a that will produce the reach of a Puma 560 robot, namely, 0.8772 m.

It is apparent that the maximum reach is independent of θ_1, the angle of rotation of the first joint, for motions about the first joint do not affect the reach. So, we lock the first joint and, in the posture of Fig. 5.1, rotate the third joint through one full

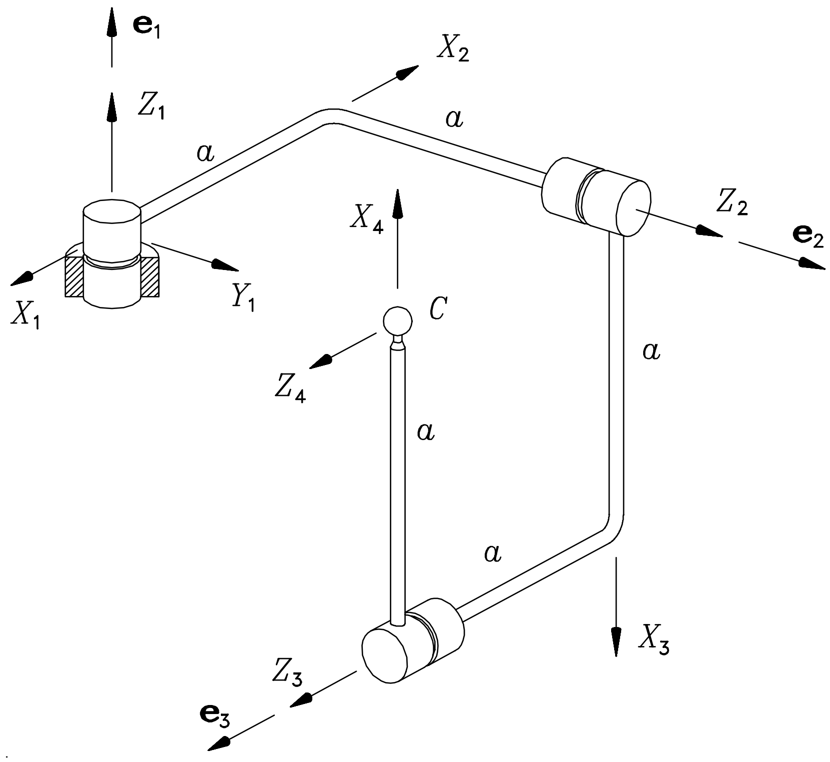


Figure 5.1: Manipulator configuration for $C(0,\, a,\, 0)$.

turn, point C thus describing a circle $\mathcal{C}$ of radius a lying in the Y_1-Z_1 plane, with centre at point O_3' of coordinates $(0,\, a,\, -a)$. Next, upon performing a full rotation of the second joint, the circle describes a toroid of axis Z_2, the problem now reducing to one of finding the point of the surface of the toroid lying the farthest from the Z_1 axis. Figure 5.2 includes side views of circle $\mathcal{C}$.

Let the trace of the toroid with the X_2-Z_2 plane be the contour $\mathcal{T}$ of Fig. 5.3. It is most convenient to represent this contour with the aid of the non-dimensional variables u and v, which are defined as

$$u \equiv \frac{x_2}{a}, \quad v \equiv \frac{z_2}{a} \tag{5.27}$$

In terms of these new variables, the equation of $\mathcal{T}$ becomes

$$\mathcal{T}: \qquad h(u,v) \equiv (u^2+v^2)(u^2+v^2-4v) - 4(u^2-v^2-1) = 0 \tag{5.28}$$

The contour $\mathcal{T}$ defined by the implicit function $h(u,v)=0$ is displayed in Fig. 5.3.

Now, the maximum distance r_M of O_1 to $\mathcal{T}$ can be found as the solution of the

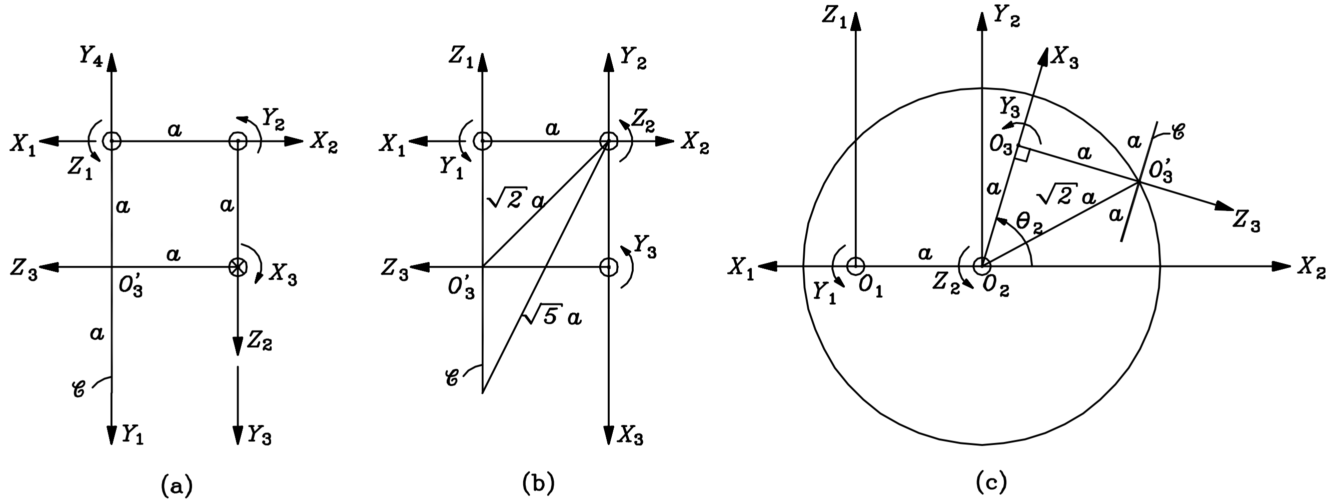


Figure 5.2: Side views of circle $\mathcal{C}$: (a) and (b) at the position of Fig. 5.1; and (c) at an arbitrary position for a given value of θ_2

optimization problem defined below:

$$f(u,v) \equiv \frac{1}{2}[(u+1)^2+v^2] \quad \to \quad \max_{u,v} \tag{5.29}$$

subject to eq.(5.28). We thus have an equality-constrained maximization problem. In order to find the normality conditions of this problem, we resort to Lagrange multipliers, thus defining a new, unconstrained, maximization problem:

$$F(u,v,\lambda) \equiv f+\lambda h \quad \to \quad \max_{u,v,\lambda} \tag{5.30}$$

The normality conditions of the foregoing problem are, thus,

$$\frac{\partial F}{\partial u} \equiv u+1+4\lambda u(u^2+v^2-2v-2)=0 \tag{5.31a}$$

$$\frac{\partial F}{\partial v} \equiv v+4\lambda(v-1)(u^2+v^2-2v)=0 \tag{5.31b}$$

$$\frac{\partial F}{\partial \lambda} \equiv (u^2+v^2)((u^2+v^2-4v)-4((u^2-v^2-1)=0 \tag{5.31c}$$

the last equation being just a restatement of the constraint, eq.(5.28). Now we eliminate λ, the Lagrange multiplier, dialytically *(Salmon, 1964) from eqs.(5.31a & b). We do this by rewriting these two equations in linear homogeneous form in the "variables" λ and 1, namely,*

$$\begin{bmatrix} 4u(u^2+v^2-2v-2) & u+1 \\ 4(v-1)(u^2+v^2-2v) & v \end{bmatrix} \begin{bmatrix} \lambda \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{5.32a}$$

Obviously, the foregoing equation requires a nontrivial solution—note that one component of the vector of "unknowns" is unity!—which in turn requires that the coefficient matrix be singular, i.e.,

$$\det\begin{bmatrix} 4u(u^2+v^2-2v-2) & u+1 \\ 4(v-1)(u^2+v^2-2v) & v \end{bmatrix} = 0 \tag{5.32b}$$

Upon expansion,

$$4u(u^2+v^2-2v-2)v - 4(v-1)(u^2+v^2-2v)(u+1) = 0$$

or

$$\mathcal{S}: \quad (u^2+v^2-2v-2)(u-v+1) - 2uv = 0 \tag{5.32c}$$

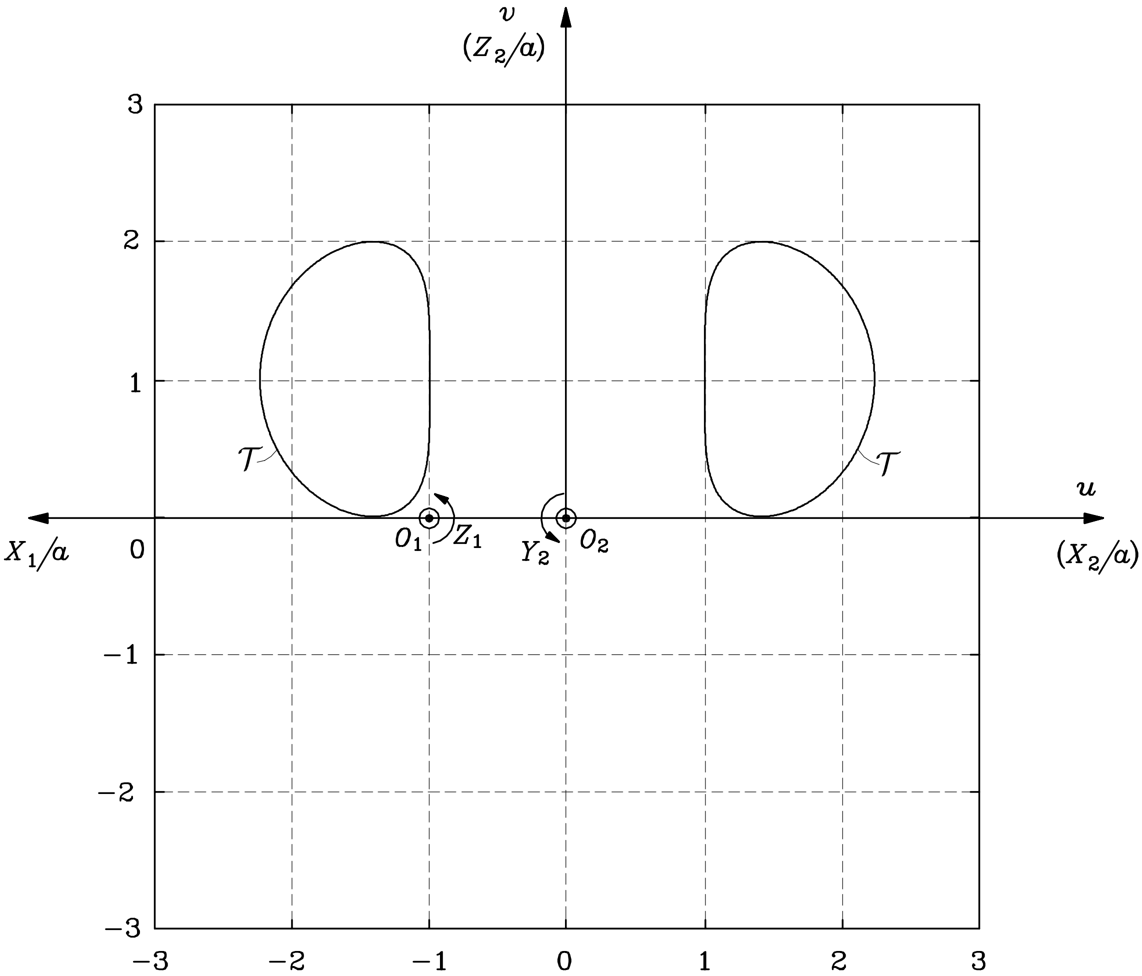


Figure 5.3: Contour of the trace $\mathcal{T}$ of the toroid on the u-v plane

Now, the maximum reach is found via the solution of the system of polynomial equations *(5.28) and (5.32c). The former is a quartic equation, the latter being* *cubic. The* Bezout number[2] *of the foregoing system of equations is defined as the product of the degrees of those equations, i.e.,* $4 \times 3 = 12$, *which gives an upper bound of 12 for the number of solutions, both real and imaginary, of the problem at hand. One graphical means of obtaining estimates of the real solutions of this system consists in plotting the two corresponding contours in the u-v plane, as shown in Fig. 5.4. The maximum reach occurs apparently, at point A, of coordinates* (2.2, 1.4) *estimated by inspection, which leads to a visual estimate of* r_M, *namely,*

$$r_m \approx 3.5a \tag{5.33}$$

The four intersections of these two curves correspond to the four stationary values of the distance from a point in the trace $\mathcal{T}$ *to the point* O_1 *in the u-v plane. Of these four intersections, two are local maxima and two local minima. The* **normality** *of* ∇f, *which in this case is identical to the vector from* O_1 *to* $\mathcal{T}$ *at the intersection points, is to be highlighted.*

The foregoing system is solved more precisely using computer algebra, thus obtaining the four real solutions given below:

$$\begin{aligned}
(u)_A &= 2.132242, & (v)_A &= 1.148990 \\
(u)_B &= -1.578095, & (v)_B &= 1.975316 \\
(u)_C &= -1.132242, & (v)_C &= 0.116796 \\
(u)_D &= 1.025308, & (v)_D &= 0.366325
\end{aligned}$$

which lead to reach values of

$$r_A = 3.459606a, \quad r_B = 2.058171a, \quad r_C = 0.176435a, \quad r_D = 2.058171a$$

for a global maximum reach of

$$r_M = 3.459606a$$

The value of a that will yield the foregoing maximum reach is thus found as

$$3.460a = 0.8772 \quad \Rightarrow \quad a = 0.2535 \text{ m}$$

[2]To define the Bezout number of a system of p polynomial equations in p variables x_1, x_2, ..., x_p, we look first at the ith equation: A typical term of this equation involves the product $x_1^{d_{1i}} x_2^{d_{2i}} \cdots x_p^{d_{pi}}$. The degree d_i of this equation is the maximum of $d_{1i} + d_{2i} + \ldots + d_{pi}$, for $i = 1, \ldots, N_i$, where N_i denotes the number of terms of the ith equation. The Bezout number N_B of this system is defined as $N_B = d_1 d_2 \ldots d_p$.

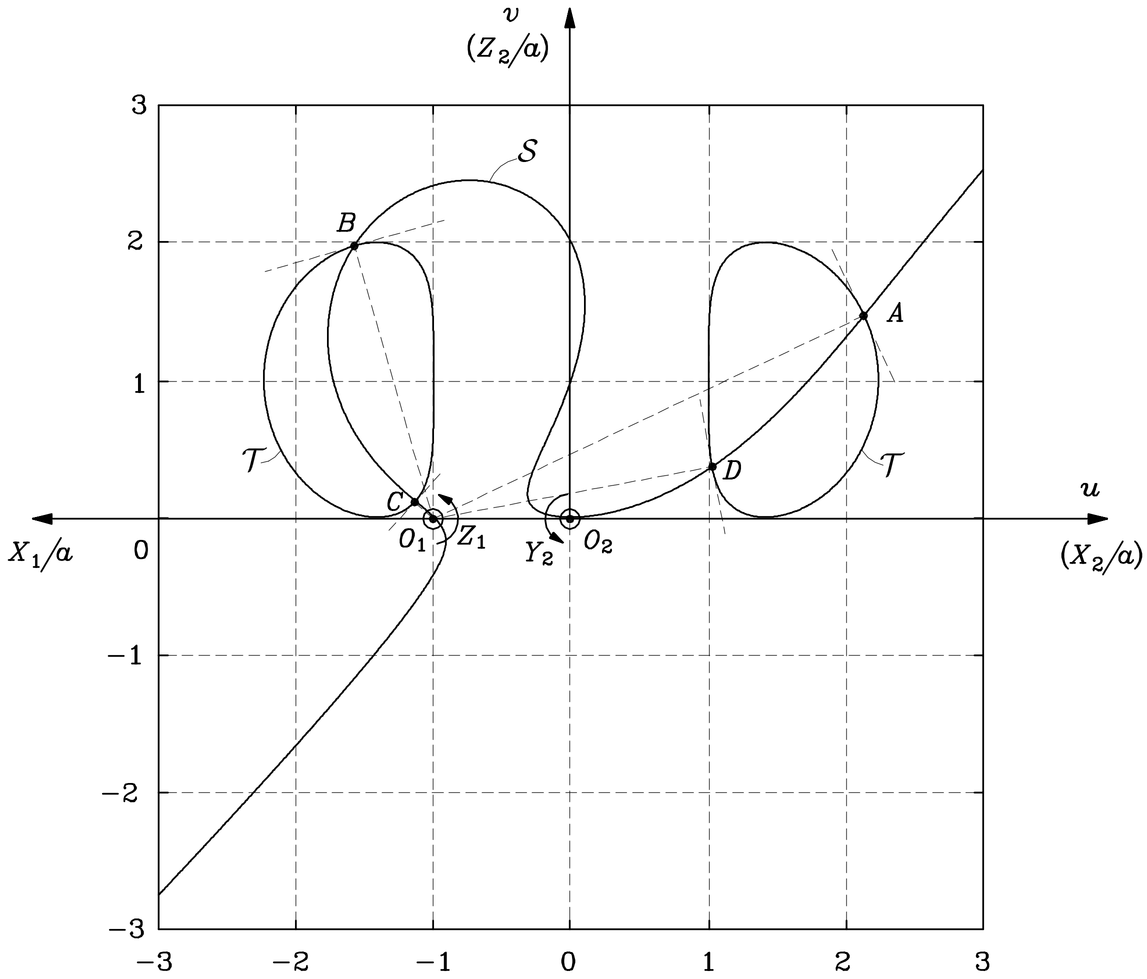


Figure 5.4: Plots of the two contours, $\mathcal{S}$ and $\mathcal{T}$

thereby completing the solution.

The verification of the first- and second-order normality conditions is left as an exercise.

Example 5.3.2 (The Equilibrium Configuration of a Four-Link Chain) *We consider here the problem of determining the equilibrium configuration of a chain composed of four identical links of length L each, suspended at two points located at the same level, a distance d apart. This problem was proposed by Luenberger (1984) to illustrate methods of nonlinear programming. Here, we use a simplified version of this problem with the purpose of obtaining a solution by simple equation-solving.*

At the outset, we exploit the symmetry of the problem, which enables us to reduce the number of design variables to only two, namely, the inclination of the two links on the left half of the chain. Let θ_i, for $i = 1, 2$, denote the angle made by the axis of the ith link from the vertical and μ denote the mass distribution per unit length, *while g represents the gravity acceleration. The potential energy V of the whole chain is, thus, for an arbitrary configuration of the chain,*

$$V(\theta_1, \theta_2) = -2\mu g L\left(\frac{1}{2}\cos\theta_1 + \cos\theta_1 + \frac{1}{2}\cos\theta_2\right)$$

which is a minimum at an equilibrium configuration. However, notice that the two design variables are not independent, for their horizontal span must be exactly $d/2$, i.e.,

$$L(\sin\theta_1 + \sin\theta_2) - \frac{d}{2} = 0$$

The optimum design problem at hand now has the form

$$f(\theta_1, \theta_2) \equiv \frac{V}{\mu g L} = -3\cos\theta_1 - \cos\theta_2 \quad \rightarrow \quad \min_{\theta_1, \theta_2}$$

subject to

$$h(\theta_1, \theta_2) = \sin\theta_1 + \sin\theta_2 - p = 0, \quad p \equiv \frac{d}{2L}$$

The Lagrangian of the problem is to be minimized, i.e.,

$$F(\theta_1, \theta_2) = -3\cos\theta_1 - \cos\theta_2 + \lambda(\sin\theta_1 + \sin\theta_2 - p) \quad \rightarrow \quad \min_{\theta_1, \theta_2, \lambda}$$

subject to no constraints. The normality conditions of the unconstrained problem are, thus

$$\frac{\partial F}{\partial \theta_1} = 3\sin\theta_1 + \lambda\cos\theta_1 = 0$$
$$\frac{\partial F}{\partial \theta_2} = \sin\theta_2 + \lambda\cos\theta_2 = 0$$
$$\frac{\partial F}{\partial \lambda} = \sin\theta_1 + \sin\theta_2 - p = 0$$

The problem has thus been reduced to solving the foregoing system of three nonlinear equations in three unknowns, θ_1, θ_2 and λ. While this nonlinear system can be solved using the Newton-Raphson method, the simplicity of the equations lends itself to a more comprehensive approach. Indeed, the Newton-Raphson method yields one single solution at a time, the user never knowing whether any other solutions exist. Moreover, there is no guarantee that the solution found is a minimum and not a maximum or a saddle point.

For starters, we can eliminate λ from the above equations, for it appears linearly in the first two of those. We thus rewrite those two equations in the form

$$\mathbf{A}\mathbf{x} = \mathbf{0}_2$$

with $\mathbf{0}_2$ *denoting the two-dimensional zero vector, while* $\mathbf{A}$ *and* $\mathbf{x}$ *are defined as*

$$\mathbf{A} \equiv \begin{bmatrix} \cos\theta_1 & 3\sin\theta_1 \\ \cos\theta_2 & \sin\theta_2 \end{bmatrix}, \quad \mathbf{x} \equiv \begin{bmatrix} \lambda \\ 1 \end{bmatrix} \neq \mathbf{0}_2$$

Since the solution sought cannot be zero, the above homogeneous system must admit a nontrivial solution, which calls for $\mathbf{A}$ *to be singular, i.e.,*

$$\Delta(\theta_1, \theta_2) \equiv \det(\mathbf{A}) = 0$$

Upon expansion,

$$\Delta(\theta_1, \theta_2) = \cos\theta_1 \sin\theta_2 - 3\sin\theta_1 \cos\theta_2 = 0$$

which we shall call the reduced normality condition. *We thus have eliminated* λ dialytically *(Salmon, 1964), the problem thus reducing to a system of two equations in two unknowns,* $h(\theta_1, \theta_2) = 0$ *and* $\Delta(\theta_1, \theta_2) = 0$. *We can further reduce the same system to one single equation in one single unknown, which can be done by dialytic elimination as well. However, notice that dialytic elimination is applicable to systems of polynomial equations, while the two equations at hand are not polynomial; they are trigonometric. Nevertheless, by application of the well-known trigonometric "half-tan" identities:*

$$\cos x \equiv \frac{1 - T^2}{1 + T^2}, \quad \sin x \equiv \frac{2T}{1 + T^2}, \quad T \equiv \tan\left(\frac{x}{2}\right)$$

the two equations can be transformed into polynomial equations. We will not pursue here this elimination procedure. Instead, we plot the two foregoing functions in the θ_1*-*θ_2 *plane, the solutions sought being found visually at the intersection of the corresponding contours. In order to plot the contours, however, we must assign a numerical value to parameter* p*. By assuming* $d = 1.25$ m *and* $L = 0.5$ m*, we obtain* $p = 1.25$*. These contours are plotted in Fig. 5.5.*

The contours apparently intersect at two points, of coordinates estimated visually at

$$\theta_1 = 0.45, \quad \theta_2 = 1.00 \quad \text{and} \quad \theta_1 = 2.70, \quad \theta_2 = 2.20$$

with all values in radians. These values are quite rough. Better values can be obtained by means of Newton-Raphson's method applied to the two nonlinear equations, using the foregoing estimates as initial guesses. Alternatively, the two equations can be solved dialytically by means of computer algebra. For example, upon invoking Maple's "`solve`*" procedure, the real roots below were reported:*

$$\theta_1 = 0.4449420670, \ \theta_2 = 0.9607027573 \quad \text{and} \quad \theta_1 = 2.696650587, \ \theta_2 = 2.180889896$$

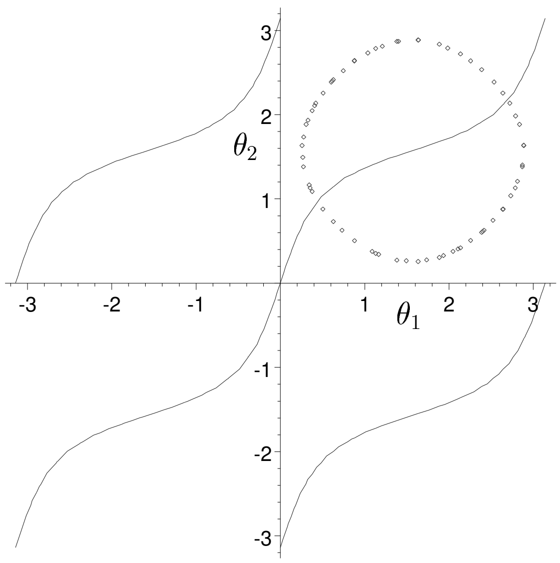


Figure 5.5: The reduced normality condition and the equality constraint (dashed contour)

Translated into degrees, the foregoing angles read:

$$\theta_1 = 25.49330256°,\ \theta_2 = 55.04421335° \text{ and } \theta_1 = 154.5066974°,\ \theta_2 = 124.9557866°$$

The first solution corresponds, apparently, to a minimum, the second to a maximum. If this is the case, then the sum of the corresponding roots for the two solutions should be π, which is the case. Moreover, upon evaluation of the objective function at the two solutions, we obtain

$$f(0.4449420670, 0.9607027573) = -1.640425478$$
$$f(2.696650587, 2.180889896) = 1.640425479$$

which clearly shows that the first solution is a maximum, the second a minimum. Notice the symmetry of the objective function at the two foregoing extrema.

The first- and second-order normality conditions should be verified numerically. The chain at its equilibrium configuration is displayed in Fig. 5.6.

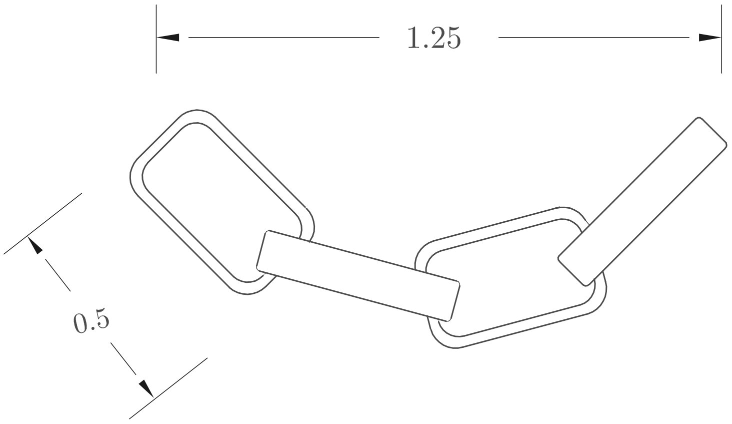


Figure 5.6: The equilibrium configuration of a four-link chain

5.4 Linear-Quadratic Problems

5.4.1 The Minimum-Norm Solution of Underdetermined Systems

We start by recalling a concept of paramount importance in optimization:

Definition 5.4.1 (Convex set) *A set of points $\mathcal{C}$ is* convex *if, given any two distinct points P_1 and P_2 of the set, then any point P of $\mathcal{C}$ comprised between P_1 and P_2 also belongs to the set. Otherwise, the set is* nonconvex.

More formally, if $\mathbf{x}_i$ denotes the position vector of P_i, for $i = 1, 2$, and $\mathbf{x}$ that of P, then, for any scalar α comprised in the interval $[\,0,\ 1\,]$, we can express the position vector of P as a *convex combination* of those of P_1 and P_2, namely,

$$\mathbf{x} = \alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2, \quad 0 \le \alpha \le 1 \tag{5.34}$$

We can thus rephrase the definition of convex set as

Definition 5.4.2 (Convex set—An alternative definition) *A set of points $\mathcal{C}$ is convex if, given any two distinct points of position vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, then the point whose position vector is a convex combination of $\mathbf{x}_1$ and $\mathbf{x}_2$ also belongs to $\mathcal{C}$.*

Germane to the concept of convex set is that defined below:

Definition 5.4.3 (convex function) *A function $f(\mathbf{x})$ is convex if, for any $\mathbf{x}_1$ and $\mathbf{x}_2$, and a $\mathbf{x}$ defined as a convex combination of $\mathbf{x}_1$ and $\mathbf{x}_2$, and given, e.g., as in* *eq.(5.34)*,

$$f(\mathbf{x}) \leq \alpha f(\mathbf{x}_1) + (1-\alpha) f(\mathbf{x}_2) \tag{5.35}$$

Now we study the *underdetermined system* of linear equations

$$\mathbf{C}\mathbf{x} = \mathbf{d} \tag{5.36}$$

where $\mathbf{C}$ is a $p \times n$ matrix with $p < n$, all equations being assumed linearly independent. Apparently, the system *admits infinitely-many solutions*. Notice that the set of solutions of this equation does not form a vector space. Indeed, since $\mathbf{0}$ is not a solution, the solution set does not include the origin, which disqualifies the set from being a vector space. However, the same set has a quite interesting property:

Fact 5.4.1 *The set of solutions of the system (5.36) is convex.*

Proof: Assume that $\mathbf{x}_1$ and $\mathbf{x}_2$ are two distinct solutions of eq.(5.36), i.e.,

$$\mathbf{C}\mathbf{x}_1 = \mathbf{d} \tag{5.37a}$$

$$\mathbf{C}\mathbf{x}_2 = \mathbf{d} \tag{5.37b}$$

Now, for a real α such that $0 \leq \alpha \leq 1$, we have

$$\mathbf{C}(\alpha\mathbf{x}_1) = \alpha\mathbf{d} \tag{5.38a}$$

$$\mathbf{C}[(1-\alpha)\mathbf{x}_2] = (1-\alpha)\mathbf{d} \tag{5.38b}$$

Upon adding sidewise eqs.(5.38a & b), we obtain

$$\mathbf{C}[\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2] = \mathbf{d} \tag{5.39}$$

thereby completing the proof.

Geometrically, eq.(5.36) represents a plane embedded in n-dimensional space, offset from the origin. Each point of the plane thus has a position vector that is a solution. Out of the infinity of solutions satisfying the equation, then, there is one that lies closest to the origin. This is the *minimum-norm solution* of eq.(5.36). We derive below this solution upon solving the problem below:

$$f(\mathbf{x}) \equiv \frac{1}{2}\|\mathbf{x}\|^2 \quad \rightarrow \quad \min_{\mathbf{x}} \tag{5.40}$$

subject to eq.(5.36). As before, we transform the above constrained problem into an unconstrained one. We do this by means of Lagrange multipliers:

$$F(\mathbf{x}) \equiv f(\mathbf{x}) + \boldsymbol{\lambda}^T(\mathbf{C}\mathbf{x} - \mathbf{d}) \quad \rightarrow \quad \min_{\mathbf{x},\boldsymbol{\lambda}} \tag{5.41}$$

subject to no constraints. The normality conditions of this problem are, thus,

$$\frac{\partial F}{\partial \mathbf{x}} \equiv \nabla f + \mathbf{C}^T\boldsymbol{\lambda} = \mathbf{x} + \mathbf{C}^T\boldsymbol{\lambda} = \mathbf{0}_n \tag{5.42a}$$

$$\frac{\partial F}{\partial \boldsymbol{\lambda}} \equiv \mathbf{C}\mathbf{x} - \mathbf{d} = \mathbf{0}_p \tag{5.42b}$$

the second set of the above equations thus being just a restatement of the system of underdetermined equations (5.36). Solving for $\mathbf{x}$ from eq.(5.42a) yields

$$\mathbf{x} = -\mathbf{C}^T\boldsymbol{\lambda} \tag{5.43a}$$

which, when substituted into eq.(5.42b), leads to

$$-\mathbf{C}\mathbf{C}^T\boldsymbol{\lambda} - \mathbf{d} = \mathbf{0}_p \tag{5.43b}$$

Since we assumed at the outset that the given eqs.(5.36) are linearly-independent, $\mathbf{C}$ is of full rank, and hence, the $p \times p$ symmetric matrix is nonsingular. As a result, this matrix is, in fact, positive-definite, the outcome being that eq.(5.43b) can be solved for $\boldsymbol{\lambda}$ by means of the Cholesky decomposition. The result is, symbolically, the minimum-norm solution $\mathbf{x}_0$ sought:

$$\mathbf{x}_0 = \mathbf{C}^\dagger\mathbf{d} \tag{5.44a}$$

where

$$\mathbf{C}^\dagger = \mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1} \tag{5.44b}$$

which is the *right Moore-Penrose generalized inverse* of the rectangular matrix $\mathbf{C}$. One can see that the straightforward evaluation of $\mathbf{C}^\dagger$ by its definition, eq. (5.44a), involves the inversion of a matrix product, which is computationally costly and prone to ill conditioning, similar to the case of the left Moore-Penrose generalized inverse of eq.(3.59b). Moreover, the solution of eq. (5.44a) does not hold when $\mathbf{C}$ is rank-deficient.

An efficient and robust alternative to computing explicitly the right Moore-Penrose generalized inverse relies in *Householder reflections*, as explained below: First, a set of $n \times n$ Householder reflections[3] $\mathbf{H}_1$, $\mathbf{H}_2$, $\ldots$, $\mathbf{H}_p$ is defined, such that

[3]See Subsection 3.5.2

the product $\mathbf{H} = \mathbf{H}_p \cdots \mathbf{H}_2\mathbf{H}_1$ transforms $\mathbf{C}^T$ into upper-triangular form, thereby obtaining

$$\mathbf{H}\mathbf{C}^T = \begin{bmatrix} \mathbf{U} \\ \mathbf{O}_{n'p} \end{bmatrix} \tag{5.45}$$

where $\mathbf{U}$ is a $p \times p$ upper-triangular matrix, which is nonsingular because we have assumed that $\mathbf{C}$ is of full rank, while $\mathbf{O}_{n'p}$ is the $n' \times p$ zero matrix, with $n' \equiv n - p$.

Further, we rewrite eq.(5.36) in the form

$$\mathbf{C}\mathbf{H}^T\mathbf{H}\mathbf{x} = \mathbf{d} \tag{5.46}$$

which does not alter the original equation (5.36) because $\mathbf{H}$ is orthogonal. Letting $\mathbf{y} = \mathbf{H}\mathbf{x}$, from eqs. (5.45) and (5.46), one can realize that $\mathbf{x}$ and $\mathbf{y}$ have the same Euclidean norm, and hence, minimizing the norm of $\mathbf{y}$ is equivalent to minimizing that of $\mathbf{x}$. Thus, $\mathbf{x}$ will be the minimum-norm solution of the underdetermined system (5.36) if $\mathbf{y}$ is, correspondingly, the minimum-norm solution of the system

$$(\mathbf{H}\mathbf{C}^T)^T\mathbf{y} = \mathbf{d} \tag{5.47a}$$

Upon substitution of eq.(5.45) into eq.(5.47a), we obtain, with a suitable partitioning of $\mathbf{y}$,

$$\begin{bmatrix} \mathbf{U}^T & \mathbf{O}_{n'p}^T \end{bmatrix} \begin{bmatrix} \mathbf{y}_U \\ \mathbf{y}_L \end{bmatrix} = \mathbf{d}, \quad \mathbf{y} \equiv \begin{bmatrix} \mathbf{y}_U \\ \mathbf{y}_L \end{bmatrix} \tag{5.47b}$$

which, upon expansion, leads to

$$\mathbf{U}^T\mathbf{y}_U + \mathbf{O}_{n'p}^T\mathbf{y}_L = \mathbf{d} \tag{5.47c}$$

whence it is apparent that $\mathbf{y}_L$ is undetermined, and hence, can be assigned *any* value, while $\mathbf{y}_U$ is determined because we have assumed that $\mathbf{C}$ is of full rank, $\mathbf{U}$ thus being nonsingular. If our intention is to minimize $\|\mathbf{x}\|$ or, equivalently, $\|\mathbf{y}\|$, whose square is given by

$$\|\mathbf{y}\|^2 = \|\mathbf{y}_U\|^2 + \|\mathbf{y}_L\|^2$$

it is apparent that the optimum choice of $\mathbf{y}_L$ is $\mathbf{y}_L = \mathbf{0}_{n'}$, with $\mathbf{0}_{n'}$ denoting the $(n-p)$-dimensional zero vector. Therefore, the minimum-norm solution $\mathbf{y}_0$ of eq.(5.47a) takes on the form:

$$\mathbf{y}_0 = \begin{bmatrix} \mathbf{U}^{-T}\mathbf{d} \\ \mathbf{0}_{n'} \end{bmatrix} \tag{5.48}$$

i.e., the last $(n-p)$ components of $\mathbf{y}_0$ are zero. In this way, $\mathbf{y}_0$ verifies eq. (5.46) and has a minimum norm. Then, the minimum-norm solution $\mathbf{x}_0$ can be readily computed as

$$\mathbf{x}_0 = \mathbf{H}^T\mathbf{y}_0 \tag{5.49}$$

The Case of a Rank-Deficient C Matrix

If $\mathbf{C}$ is rank-deficient, with $\text{rank}(\mathbf{C}) = r < p$, then we can proceed as described above with only r Householder reflections, namely, $\mathbf{H} = \mathbf{H}_r\mathbf{H}_{r-1}\ldots\mathbf{H}_1$, such that

$$\mathbf{HC}^T = \begin{bmatrix} \overline{\mathbf{U}} \\ \overline{\mathbf{O}} \end{bmatrix}, \quad \mathbf{y} \equiv \mathbf{Hx} \tag{5.50}$$

where $\overline{\mathbf{U}}$ is a full-rank $r \times p$ matrix with zero entries in its lower-left "corner"—this matrix has an upper-trapezoidal form—and $\overline{\mathbf{O}}$ defined as the $(n-r)\times p$ zero matrix.

Note that, in general, the rank of $\mathbf{C}$ is not known in advance. It is first learned when the p Householder reflections introduced above are defined to bring $\mathbf{C}^T$ into upper-triangular form. In the presence of a rank-deficient matrix $\mathbf{C}$, of rank $r < p$, The last $n - r$ rows of $\mathbf{HC}^T$ are all zero, and the last $p - r$ $\mathbf{H}_i$ matrices are all identical.

Upon application of the foregoing r Householder reflections, eq. (5.47b) becomes

$$[\,\overline{\mathbf{U}}^T \quad \overline{\mathbf{O}}\,] \begin{bmatrix} \overline{\mathbf{y}}_U \\ \overline{\mathbf{y}}_L \end{bmatrix} = \mathbf{d}, \quad \mathbf{y} \equiv \begin{bmatrix} \overline{\mathbf{y}}_U \\ \overline{\mathbf{y}}_L \end{bmatrix} \tag{5.51}$$

whence,

$$\overline{\mathbf{U}}^T\overline{\mathbf{y}}_U + \overline{\mathbf{O}}\overline{\mathbf{y}}_L = \mathbf{d} \tag{5.52}$$

Apparently, $\|\mathbf{x}\| = \|\mathbf{y}\|$, and hence, upon minimizing one norm, one minimizes the other one as well. Moreover,

$$\|\mathbf{y}\|^2 = \|\overline{\mathbf{y}}_U\|^2 + \|\overline{\mathbf{y}}_L\|^2$$

Therefore, the optimum choice of $\mathbf{y}$ is the one for which $\overline{\mathbf{y}}_L = \mathbf{0}_{n''}$, with $\mathbf{0}_{n''}$ denoting the n''-dimensional zero vector, and $n'' \equiv n - r$, eq.(5.52) thus reducing to

$$\overline{\mathbf{U}}^T\overline{\mathbf{y}}_U = \mathbf{d} \tag{5.53}$$

where $\overline{\mathbf{U}}^T$ is a $p \times r$ matrix with zero entries in its upper corner, i.e., this matrix has the form

$$\overline{\mathbf{U}}^T = \begin{bmatrix} \overline{\mathbf{L}} \\ \mathbf{M} \end{bmatrix} \tag{5.54}$$

in which $\overline{\mathbf{L}}$ is a nonsingular $r \times r$ lower-triangular matrix and $\mathbf{M}$ is a $(p - r) \times r$ matrix. Moreover, since $\overline{\mathbf{U}}$ has been assumed of full rank, $\overline{\mathbf{U}}^T$ is also of full rank, its last $p - r$ rows being linearly dependent from its first r rows. That is, the $p - r$ rows of $\mathbf{M}$ are linearly dependent from the r rows of $\overline{\mathbf{L}}$. This means that $\overline{\mathbf{y}}_U$ is determined from the first r equations of eq.(5.53). We can thus use only those equations, which are, moreover, in lower-triangular form already, to compute $\mathbf{y}_U$ by *forward substitution*. *Symbolically*, then, we have

$$\overline{\mathbf{y}}_U = \overline{\mathbf{L}}^{-1}\mathbf{d}, \quad \mathbf{x}_0 = \mathbf{H}^T \begin{bmatrix} \overline{\mathbf{L}}^{-1}\mathbf{d} \\ \mathbf{0}_{n''} \end{bmatrix} \tag{5.55}$$

Alternatively, and if CPU time is not an issue, we can use all redundant scalar equations of that vector equation. We do this, then, by application of another set of r Householder reflections, $\overline{\mathbf{H}}_1, \overline{\mathbf{H}}_2, \ldots, \overline{\mathbf{H}}_r$, thereby obtaining

$$\overline{\mathbf{H}}\mathbf{U}^T\mathbf{y}_U = \overline{\mathbf{H}}\mathbf{d}, \quad \overline{\mathbf{H}} \equiv \overline{\mathbf{H}}_1\overline{\mathbf{H}}_2 \ldots \overline{\mathbf{H}}_r \tag{5.56}$$

whence the optimum solution is obtained in the way explained for overdetermined systems in Section 3.5. The details are left as an exercise.

Example 5.4.1 (The Solution of $\mathbf{a} \times \mathbf{x} = \mathbf{b}$)
Let $\mathbf{a}$, $\mathbf{b}$, *and* $\mathbf{x}$ *be three 3-dimensional Cartesian vectors. We would like to solve the equation*

$$\mathbf{a} \times \mathbf{x} = \mathbf{b}$$

for $\mathbf{x}$. *It is well known, however, that the foregoing equation contains only two independent scalar equations, which prevents us from finding "the* $\mathbf{x}$*" that verifies that equation. Thus, we can proceed by finding a specific* $\mathbf{x}$, $\mathbf{x}_0$, *that verifies* any *two of these three equations and that is of minimum norm. To this end, we expand that equation into its three components:*

$$\begin{aligned} a_2x_3 - a_3x_2 &= b_1 \\ a_3x_1 - a_1x_3 &= b_2 \\ a_1x_2 - a_2x_1 &= b_3 \end{aligned}$$

Note that the foregoing equation can be cast in the form of eq.(3.33) if we define matrix $\mathbf{A}$ *as*

$$\mathbf{A} \equiv \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

which is apparently skew-symmetric, i.e.,

$$\mathbf{A}^T = -\mathbf{A}$$

In fact, $\mathbf{A}$ is the cross-product matrix of $\mathbf{a}$. Picking up, for example, the first two scalar equations above, we obtain an underdetermined system of the form (5.36), with

$$\mathbf{C} \equiv \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

and hence, the corresponding minimum-norm solution $\mathbf{x}_0$ is given by eqs.(5.44a & b), with

$$\mathbf{C}\mathbf{C}^T = \begin{bmatrix} a_2^2 + a_3^2 & -a_1 a_2 \\ -a_1 a_2 & a_1^2 + a_3^2 \end{bmatrix}$$

Hence,

$$(\mathbf{C}\mathbf{C}^T)^{-1} = \frac{1}{\Delta} \begin{bmatrix} a_1^2 + a_3^2 & a_1 a_2 \\ a_1 a_2 & a_2^2 + a_3^2 \end{bmatrix}$$

where

$$\Delta \equiv \det(\mathbf{C}\mathbf{C}^T) = (a_2^2 + a_3^2)(a_1^2 + a_3^2) - a_1^2 a_2^2 > 0$$

a relation that the reader can readily prove. Therefore,

$$\mathbf{C}^\dagger = \frac{1}{\Delta} \begin{bmatrix} a_1 a_2 a_3 & (a_2^2 + a_3^2) a_3 \\ -(a_1^2 + a_3^2) a_3 & -a_1 a_2 a_3 \\ a_2 a_3^2 & -a_1 a_3^2 \end{bmatrix}$$

and

$$\mathbf{x}_0 = \frac{1}{\Delta} \begin{bmatrix} a_1 a_2 a_3 b_1 + (a_2^2 + a_3^2) a_3 b_2 \\ -(a_1^2 + a_3^2) a_3 b_1 - a_1 a_2 a_3 b_2 \\ a_2 a_3^2 b_1 - a_1 a_3^2 b_2 \end{bmatrix}$$

Notice that the foregoing solution depends on the condition $a_3 \neq 0$. If $a_3 = 0$, or very close to 0, then $\mathbf{C}$ becomes either ill-conditioned or rank-deficient, which is bad news. Apparently, the foregoing solution has an element of arbitrariness that may lead either to ill-conditioning or to rank-deficiency. There is no guarantee that the two equations chosen are the best choice from the condition-number viewpoint. Besides, that approach leaves aside useful information, that of the deleted equation. The alternative approach uses all three equations, to which one fourth equation is adjoined, namely, the minimum-norm condition, as described below.

First we observe that, if $\mathbf{x}$ has been found that verifies the given cross-product equation, then any other vector $\mathbf{x} + \alpha\mathbf{a}$, for $\alpha \in \mathbb{R}$, verifies that equation. Apparently, then, the minimum-norm $\mathbf{x}$ is that whose component along $\mathbf{a}$ vanishes, i.e.,

$$\mathbf{a}^T \mathbf{x} = 0$$

Upon adjoining the foregoing equation to the original three, we end up with an apparently overdetermined system of four equations with three unknowns, of the form

$$\mathbf{M}\mathbf{x} = \mathbf{n}$$

where $\mathbf{M}$ *and* $\mathbf{n}$ *are given by*

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} \\ \mathbf{a}^T \end{bmatrix}, \quad \mathbf{n} = \begin{bmatrix} \mathbf{b} \\ 0 \end{bmatrix}$$

Hence, $\mathbf{M}$ *is a* 4×3 *matrix, while* $\mathbf{n}$ *is a 4-dimensional vector. The least-square approximation of the new system is, then, the minimum-norm solution of the original system, provided the latter is verified exactly, which it is, as will become apparent. Indeed, the least-square approximation of the new system takes the form*

$$\mathbf{x}_L = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{n} \tag{5.58}$$

While we have strongly advised against the explicit computation of generalized inverses, our advice is valid only as pertaining to numerical computations. In the case at hand, we will pursue not a numerical, but rather a symbolic *computation of the solution sought.*

The first issue now is whether $\mathbf{M}^T\mathbf{M}$ *is invertible, but it is so and, moreover, its inverse is extremely simple to find:*

$$\mathbf{M}^T\mathbf{M} = [\,\mathbf{A}^T \quad \mathbf{a}\,]\begin{bmatrix} \mathbf{A} \\ \mathbf{a}^T \end{bmatrix} = \mathbf{A}^T\mathbf{A} + \mathbf{a}\mathbf{a}^T$$

But, since $\mathbf{A}$ *is skew-symmetric,*

$$\mathbf{M}^T\mathbf{M} = -\mathbf{A}^2 + \mathbf{a}\mathbf{a}^T$$

as the reader can readily verify; moreover,

$$\mathbf{A}^2 = -\|\mathbf{a}\|^2\mathbf{1} + \mathbf{a}\mathbf{a}^T$$

Hence,

$$\mathbf{M}^T\mathbf{M} = \|\mathbf{a}\|^2\mathbf{1}$$

which means that $\mathbf{M}$ *is* isotropic, *i.e., optimally-conditioned. Therefore,*

$$(\mathbf{M}^T\mathbf{M})^{-1} = \frac{1}{\|\mathbf{a}\|^2}\mathbf{1}$$

That is ,

$$\mathbf{x}_L = \frac{1}{\|\mathbf{a}\|^2}\mathbf{1}\,[\,\mathbf{A}^T \quad \mathbf{a}\,]\begin{bmatrix}\mathbf{b}\\ 0\end{bmatrix} = \frac{1}{\|\mathbf{a}\|^2}\mathbf{A}^T\mathbf{b}$$

which can be further expressed as

$$\mathbf{x}_L = -\frac{\mathbf{a}\times\mathbf{b}}{\|\mathbf{a}\|^2} \tag{5.59}$$

thereby obtaining a much simpler, and robust, expression than that displayed above as $\mathbf{x}_0$.

5.4.2 Least-Square Problems Subject to Linear Constraints

Given the system of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{5.60}$$

where $\mathbf{A}$ is a full-rank $q \times n$ matrix, with $q > n$, and $\mathbf{b}$ is a q-dimensional vector, find an n-dimensional vector $\mathbf{x}$ that verifies the above system with the least-square error, subject to the linear equality constraints

$$\mathbf{C}\mathbf{x} = \mathbf{d} \tag{5.61}$$

with $\mathbf{C}$ a full-rank $p \times n$ matrix and $\mathbf{d}$ a p-dimensional vector. Moreover, $\mathbf{W}$ is a $q \times q$ positive-definite weighting matrix, with q, p and n subject to

$$q + p > n \quad \text{and} \quad n > p \tag{5.62}$$

The least-square error of eqs.(5.60) is defined as

$$f \equiv \frac{1}{2}(\mathbf{A}\mathbf{x} - \mathbf{b})^T\mathbf{W}(\mathbf{A}\mathbf{x} - \mathbf{b}) \tag{5.63}$$

As usual, we solve this problem by introducing Lagrange multipliers:

$$F(\mathbf{x};\boldsymbol{\lambda}) \equiv f(\mathbf{x}) + \boldsymbol{\lambda}^T(\mathbf{C}\mathbf{x} - \mathbf{d}) \quad \min_{\mathbf{x},\boldsymbol{\lambda}} \tag{5.64}$$

subject to no constraints.

The first-order normality conditions of the foregoing problem are

$$\frac{\partial F}{\partial \mathbf{x}} \equiv \mathbf{A}^T\mathbf{W}(\mathbf{A}\mathbf{x} - \mathbf{b}) + \mathbf{C}^T\boldsymbol{\lambda} = \mathbf{0}_n \tag{5.65a}$$

$$\frac{\partial F}{\partial \boldsymbol{\lambda}} \equiv \mathbf{C}\mathbf{x} - \mathbf{d} = \mathbf{0}_p \tag{5.65b}$$

Since $\mathbf{A}$ is assumed of full rank and $\mathbf{W}$ is positive-definite, we can solve eq.(5.65a) for $\mathbf{x}$ in terms of $\boldsymbol{\lambda}$, namely,

$$\mathbf{x} = (\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}(\mathbf{A}^T\mathbf{W}\mathbf{b} - \mathbf{C}^T\boldsymbol{\lambda}) \tag{5.66}$$

Upon substituting the above expression into eq.(5.65b), we obtain

$$\mathbf{C}(\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{C}^T\boldsymbol{\lambda} = \mathbf{C}(\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}\mathbf{b} - \mathbf{d}$$

whence,

$$\boldsymbol{\lambda} = [\mathbf{C}(\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{C}^T]^{-1}[\mathbf{C}(\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}\mathbf{b} - \mathbf{d}] \tag{5.67}$$

Now, the foregoing expression for $\boldsymbol{\lambda}$ is substituted, in turn, into eq.(5.66), thereby obtaining the optimum value of $\mathbf{x}$, $\mathbf{x}_0$, namely,

$$\mathbf{x}_0 = \mathbf{P}\mathbf{Q}\mathbf{b} + \mathbf{R}\mathbf{d} \tag{5.68a}$$

where $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{R}$ are the $n \times n$-, $n \times m$- and $n \times p$ matrices given below:

$$\mathbf{P} = \mathbf{1}_n - \mathbf{R}\mathbf{C} \tag{5.68b}$$

$$\mathbf{Q} = (\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W} \tag{5.68c}$$

$$\mathbf{R} = (\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{C}^T[\mathbf{C}(\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{C}^T]^{-1} \tag{5.68d}$$

with $\mathbf{1}_n$ standing for the $n \times n$ identity matrix. The solution derived above, while being exact, for it is symbolic, is unsuitable for numerical implementation. Indeed, this solution contains inversions of products of several matrices times their transposes, which brings about ill-conditioning. Various approaches to the numerical solution of this problem will be studied in Ch. 6.

5.5 Equality-Constrained Nonlinear Least Squares

We consider here the problem of finding the least-square error f of an overdetermined system of nonlinear equations, namely,

$$\boldsymbol{\phi}(\mathbf{x}) = \mathbf{0} \tag{5.69a}$$

subject to the nonlinear constraints

$$\mathbf{h}(\mathbf{x}) = \mathbf{0} \tag{5.69b}$$

In general, moreover, the various scalar equations of eq.(5.69a) have different relevance and are, hence, assigned different weights, which then leads to a problem of *weighted least squares*, namely,

$$f(\mathbf{x}) = \frac{1}{2}\boldsymbol{\phi}^T\mathbf{W}\boldsymbol{\phi} \quad \rightarrow \quad \min_{\mathbf{x}} \tag{5.70}$$

subject to eq.(5.69b).

In the foregoing problem, $\boldsymbol{\phi}$ and $\mathbf{x}$ are q- and n-dimensional vectors, respectively, with $q > n$, and $\mathbf{W}$ is a $q \times q$ positive-definite weighting matrix. Moreover, $\mathbf{h}$ is a l-dimensional vector of nonlinear constraints.

The normality conditions of the problem at hand are derived directly from those of the general equality-constrained problem, namely, eq.(5.12) or its dual counterpart, eq.(5.20). In our case,

$$\nabla f = \left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x}}\right)^T \frac{\partial f}{\partial \boldsymbol{\phi}} \tag{5.71a}$$

where

$$\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x}} \equiv \boldsymbol{\Phi}(\mathbf{x}), \quad \frac{\partial f}{\partial \boldsymbol{\phi}} = \mathbf{W}\boldsymbol{\phi}(\mathbf{x}) \tag{5.71b}$$

i.e., $\boldsymbol{\Phi}(\mathbf{x})$ denotes the *Jacobian matrix* of $\boldsymbol{\phi}(\mathbf{x})$ with respect to $\mathbf{x}$. Hence,

$$\nabla f = \boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\phi} \tag{5.71c}$$

where we have dispensed with the argument $\mathbf{x}$ for simplicity.

The normality condition (5.12) thus reduces to

$$[\mathbf{1} - \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{J}]\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\phi} = \mathbf{0}_n \tag{5.72}$$

What this condition states is that, at a stationary point, $\boldsymbol{\phi}$, or $\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\phi}$ for that matter, need not vanish; only the projection of the latter onto the nullspace of the gradient of the constraints must vanish.

The dual form of the same normality conditions, in turn, reduces to

$$\mathbf{L}^T\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\phi} = \mathbf{0}_{n'} \tag{5.73}$$

with $\mathbf{L}$ indicating an $n \times (n-l)$ orthogonal complement of $\mathbf{J}$, as defined in eq.(5.18).

The second-order normality conditions are now derived by assuming that we have found a stationary value of the design-variable vector, $\mathbf{x}_0$. This means that

$$\mathbf{L}^T\boldsymbol{\Phi}^T(\mathbf{x}_0)\mathbf{W}\boldsymbol{\phi}(\mathbf{x}_0) = \mathbf{0}_{n'} \tag{5.74}$$

Next, we impose the condition that, for any *feasible move* $\Delta\mathbf{x}$, the corresponding increment of f, Δf, be positive. A feasible move is obtained by resorting to the orthogonal complement $\mathbf{L}$, namely,

$$\Delta\mathbf{x} = \mathbf{L}\Delta\mathbf{u} \tag{5.75}$$

On the other hand, upon expansion of $\boldsymbol{\phi}(\mathbf{x}_0 + \Delta\mathbf{x})$ to a first order,

$$\begin{aligned}
f(\mathbf{x}_0 + \Delta\mathbf{x}) &\approx \frac{1}{2}(\boldsymbol{\phi} + \Delta\boldsymbol{\phi})^T\mathbf{W}(\boldsymbol{\phi} + \Delta\boldsymbol{\phi}) \\
&= \underbrace{\frac{1}{2}\boldsymbol{\phi}^T\mathbf{W}\boldsymbol{\phi}}_{f(\mathbf{x}_0)} + \frac{1}{2}(\boldsymbol{\phi}^T\mathbf{W}\Delta\boldsymbol{\phi} + \Delta\boldsymbol{\phi}^T\mathbf{W}\boldsymbol{\phi}) + \frac{1}{2}\Delta\boldsymbol{\phi}^T\mathbf{W}\Delta\boldsymbol{\phi}
\end{aligned}$$

whence,

$$\Delta f \equiv f(\mathbf{x}_0 + \Delta\mathbf{x}) - f(\mathbf{x}_0) \approx \frac{1}{2}(\boldsymbol{\phi}^T\mathbf{W}\Delta\boldsymbol{\phi} + \Delta\boldsymbol{\phi}^T\mathbf{W}\boldsymbol{\phi}) + \frac{1}{2}\Delta\boldsymbol{\phi}^T\mathbf{W}\Delta\boldsymbol{\phi} \tag{5.76a}$$

where

$$\Delta\boldsymbol{\phi} = \boldsymbol{\Phi}\Delta\mathbf{x} = \boldsymbol{\Phi}\mathbf{L}\Delta\mathbf{u} \tag{5.76b}$$

Therefore,

$$\begin{aligned}
\Delta f &= \frac{1}{2}[\underbrace{\boldsymbol{\phi}^T(\mathbf{x}_0)\mathbf{W}\boldsymbol{\Phi}(\mathbf{x}_0)\mathbf{L}}_{=\mathbf{0}_{n-l}^T}\Delta\mathbf{u} + \Delta\mathbf{u}^T\underbrace{\mathbf{L}^T\boldsymbol{\Phi}^T(\mathbf{x}_0)\mathbf{W}\boldsymbol{\phi}(\mathbf{x}_0)}_{=\mathbf{0}_{n-l}}] \\
&\quad + \frac{1}{2}\Delta\mathbf{u}^T\mathbf{L}^T\boldsymbol{\Phi}^T(\mathbf{x}_0)\mathbf{W}\boldsymbol{\Phi}\mathbf{L}\Delta\mathbf{u} \\
&= \frac{1}{2}\Delta\mathbf{u}^T\mathbf{L}^T\boldsymbol{\Phi}^T(\mathbf{x}_0)\mathbf{W}\boldsymbol{\Phi}(\mathbf{x}_0)\mathbf{L}\Delta\mathbf{u} > 0
\end{aligned} \tag{5.76c}$$

the terms inside the brackets vanishing by virtue of the first-order normality conditions. What condition (5.76c) states is that, for the stationary value $\mathbf{x}_0$ to be a minimum, the feasible Hessian $\mathbf{L}^T\boldsymbol{\Phi}^T(\mathbf{x}_0)\mathbf{W}\boldsymbol{\Phi}(\mathbf{x}_0)\mathbf{L}$ must be positive-definite. However, at the outset we defined $\mathbf{W}$ as positive-definite, while $\mathbf{L}$ is of full rank. Hence, the feasible Hessian is necessarily positive-definite, except for points at which $\boldsymbol{\Phi}$ becomes rank-deficient, at which the said Hessian becomes positive-semidefinite. As a consequence, then,

Fact 5.5.1 *To a first-order approximation of $\boldsymbol{\phi}(\mathbf{x})$, a stationary point of the weighted least-square approximation of the overdetermined nonlinear system $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{0}$ is a minimum or a saddle point; never a maximum.*

5.6 Linear Least-Square Problems Under Quadratic Constraints

An important family of design problems lends itself to a formulation whereby the objective function is quadratic in a linear function of the design vector $\mathbf{x}$, while the constraints are quadratic in $\mathbf{x}$. Contrary to the case of linear least-squares subject to linear constraints, this family of problems does not allow, in general, for closed-form solutions, the reason being that their normal equations are nonlinear. Let us consider

$$f(\mathbf{x}) \equiv \frac{1}{2}(\mathbf{b} - \mathbf{A}\mathbf{x})^T \mathbf{W}(\mathbf{b} - \mathbf{A}\mathbf{x}) \quad \rightarrow \quad \min_{\mathbf{x}} \tag{5.77a}$$

subject to

$$\mathbf{h}(\mathbf{x}) = \mathbf{0}_l \tag{5.77b}$$

where $\mathbf{A}$ is a $q \times n$ full-rank matrix, with $q > n$, $\mathbf{W}$ is a $q \times q$ positive-definite weighting matrix, while $\mathbf{h}$, $\mathbf{x}$ and $\mathbf{b}$ are l-, n- and q-dimensional vectors, respectively, with

$$q + l > n, \quad n > l \tag{5.77c}$$

Moreover, in this particular case, the ith component of vector $\mathbf{h}$ is quadratic, namely,

$$h_i(\mathbf{x}) \equiv \frac{1}{2}\mathbf{x}^T \mathbf{P}_i \mathbf{x} + \mathbf{q}_i + r_i \tag{5.77d}$$

in which $\mathbf{P}_i$ is a known $n \times n$ symmetric matrix, while $\mathbf{q}_i$ is a n-dimensional given vector and r_i is a given scalar. Apparently, then, the ith row of $\mathbf{J}$, the Jacobian of $\mathbf{h}$ with respect to $\mathbf{x}$, takes the form

$$\frac{\partial h_i}{\partial \mathbf{x}} = \mathbf{P}_i \mathbf{x} + \mathbf{q}_i \tag{5.78}$$

whence $\mathbf{J}$ is *linear* in $\mathbf{x}$. The first-order normality conditions (5.12) now take the form

$$[\mathbf{1}_n - \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{J}]\mathbf{A}^T \mathbf{W}(\mathbf{b} - \mathbf{A}\mathbf{x}) = \mathbf{0}_n \tag{5.79}$$

It is thus apparent that, although $\mathbf{J}$ is linear in $\mathbf{x}$, the normality conditions are polynomial, thereby leading to a problem lacking a closed-form solution, except for special cases, like the one included below.

Example 5.6.1 (A Quadratic Objective Function with a Quadratic Constraint)

Consider an optimization problem with an objective function defined as

$$f(\mathbf{x}) = \frac{1}{2}(9x_1^2 - 8x_1x_2 + 3x_2^2) \quad \rightarrow \quad \min_{x_1, x_2}$$

subject to the quadratic condition

$$h(\mathbf{x}) = x_1^2 + x_2^2 - 1 = 0 \tag{5.80}$$

The objective function $f(\mathbf{x})$ can be factored as

$$f(\mathbf{x}) = \frac{1}{2}(\mathbf{b} - \mathbf{A}\mathbf{x})^T\mathbf{W}(\mathbf{b} - \mathbf{A}\mathbf{x})$$

with

$$\mathbf{A} = \mathbf{1}_2, \quad \mathbf{b} = \mathbf{0}_2, \quad \mathbf{W} = \begin{bmatrix} 9 & -4 \\ -4 & 3 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{5.81}$$

Solution: *Upon adjoining the constraint to the objective function, we obtain the Lagrangian $F(\mathbf{x};\boldsymbol{\lambda})$, namely,*

$$F(\mathbf{x};\boldsymbol{\lambda}) = \frac{1}{2}(9x_1^2 - 8x_1x_2 + 3x_2^2) + \lambda(x_1^2 + x_2^2 - 1)$$

that we want to minimize subject to no constraints. The normality conditions are, thus

$$\begin{aligned} \frac{\partial F}{\partial x_1} &= 9x_1 - 4x_2 + 2\lambda x_1 = 0 \\ \frac{\partial F}{\partial x_2} &= -4x_1 + 3x_2 + 2\lambda x_2 = 0 \\ \frac{\partial F}{\partial \lambda} &= x_1^2 + x_2^2 - 1 = 0 \end{aligned}$$

We can now eliminate λ from the first and the second of the above equations. We do this dialytically, i.e., we write these two equations in linear homogeneous form in λ and 1, i.e.,

$$\mathbf{M}\mathbf{y} = \mathbf{0}_2$$

where

$$\mathbf{M} = \begin{bmatrix} 2x_1 & 9x_1 - 4x_2 \\ 2x_2 & -4x_1 + 3x_2 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \lambda \\ 1 \end{bmatrix} \neq \mathbf{0}_2$$

with $\mathbf{0}_2$ denoting the 2-dimensional zero vector. Now, the above linear homogeneous equation in $\mathbf{y}$ cannot be zero, for $\mathbf{y} \neq \mathbf{0}_2$, and hence, matrix $\mathbf{M}$ must be singular, which is stated as

$$\det(\mathbf{M}) = 0$$

Upon expansion, the foregoing equation leads to

$$\det(\mathbf{M}) = 2x_1(-4x_1 + 3x_2) - 2x_2(9x_1 - 4x_2) = 0$$

or, after simplification,

$$x_1^2 + \frac{3}{2}x_1x_2 - x_2^2 = 0$$

thereby reducing the problem to the solution of two quadratic equations in two unknowns, the above equation and the third normality condition. Upon eliminating of x_2 *from the latter, and substitution of the expression thus resulting into the remaining equation, we obtain, after some simplifications,*

$$x_1^4 - x_1^2 + \frac{4}{25} = 0$$

which is, in fact, a quadratic equation in x_1^2*, its roots being*

$$(x_1^2)_{1,2} = \frac{1}{5}, \quad \frac{4}{5}$$

whence the four roots follow:

$$(x_1)_{1,2} = \pm\frac{\sqrt{5}}{5}, \quad (x_2)_{3,4} = \pm\frac{2\sqrt{5}}{5}$$

which yields, correspondingly,

$$(x_2)_{1,2} = \pm\frac{2\sqrt{5}}{5}, \quad (x_2)_{3,4} = \pm\frac{\sqrt{5}}{5}$$

More general problems of this family can be solved using the methods discussed in Ch. 6 for arbitrary objective functions subject to nonlinear equality constraints.

Chapter 6

Equality-Constrained Optimization: The Orthogonal-Decomposition Algorithm

6.1 Introduction

The numerical solution of equality-constrained problems is the subject of this chapter. In this vein, we focus on methods stemming from nonlinear least-square problems, that lead to what is known as *sequential quadratic programming* (SQP). SQP appears to be the most commonly used and reliable method in nonlinear programming, for it is well suited for the solution of nonlinear programming problems, as reported by (Murray, 1997) and (Lalee et al., 1998). In SQP, the nonlinear optimization problem is approximated by a sequence of quadratic programs (QP), each being a sub-quadratic program (sub-QP).

Motivated by nonlinear least-square problems, QP works under the assumption that the feasible Hessian is positive-definite at each iteration. If this is not the case at a given iteration, then the Hessian matrix is modified to render it so, a procedure called *Hessian-stabilization*. Then, the sub-QP is solved by means of any algorithm suitable for QP problems. The procedure is terminated when a criterion is met with a prescribed tolerance. The solution procedure of SQP thus involves two phases: the stabilization of the Hessian matrix and the QP solution. The Hessian matrix can be stabilized by methods such as that proposed by Broyden (1970), Fletcher (1970), Goldfarb (1970), and Shanno (1970), which is known as the BFGS method. The BFGS method, implemented in the Matlab Optimization Toolbox, is thought to be very effective for use in general applications and thus, appears to be the most popular. In addition, methods for solving QP problems are for example, the coordinate-ascent method (Bertsekas, 1995) and quasi-Newton methods (Rao, 1996).

The main item introduced in this chapter is the *orthogonal-decomposition algorithm* (ODA), which is derived first in the context of equality-constrained linear least-square problems; then, it is applied to equality-constrained nonlinear least-square problems. Several numerical techniques, such as Householder reflections, Cholesky decomposition, the Newton-Gauss method, etc., are applied in order to obtain numerical solutions by means of procedures that are both *efficient* and *robust*. What we mean by the former is procedures that use as few floating-point operations (flops) as possible; by the latter we mean procedures that keep the roundoff error in the solution as low as possible with respect to that of the data, an item that falls in the realm of *numerical conditioning*.

The orthogonal-decomposition algorithm is implemented in a C library of routines, called ODA, in combination with *Gerschgorin stabilization* (Teng and Angeles, 2001) for arbitrary objective functions, in the framework of sequential quadratic programming (SQP). Gerschgorin stabilization is based on the Gerschgorin Theorem (Varga, 2000), which provides a region of the complex plane in which the eigenvalues of an arbitrary $n \times n$ matrix are bound to lie. Moreover, the ODA is applied in solving the underlying sub-QP.

6.2 Linear Least-Square Problems Subject to Equality Constraints: The ODA

We recall below the linear least-square problem subject to linear equality constraints: Given the overdetermined system of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{6.1}$$

find a vector $\mathbf{x}$ that verifies the above system with the least-square error, which is defined as

$$f \equiv \frac{1}{2}(\mathbf{A}\mathbf{x} - \mathbf{b})^T \mathbf{W}(\mathbf{A}\mathbf{x} - \mathbf{b}) \rightarrow \min_{\mathbf{x}} \tag{6.2}$$

subject to the linear constraints

$$\mathbf{Cx} = \mathbf{d} \tag{6.3}$$

Here, $\mathbf{x}$ is the n-dimensional vector of design variables, while $\mathbf{A}$ and $\mathbf{C}$ are $q \times n$ and $p \times n$ matrices, while $\mathbf{b}$ and $\mathbf{d}$ are q- and p-dimensional vectors. Moreover, $\mathbf{W}$ is a $q \times q$ positive-definite weighting matrix, with q, p and n subject to

$$q > n \quad \text{and} \quad p < n \tag{6.4}$$

Note that the first of the foregoing inequalities excludes the possibility of a unique solution upon solving for $\mathbf{x}$ from eq.(6.1), the second preventing a unique solution from eq.(6.3).

If $\mathbf{A}$ and $\mathbf{C}$ are full-rank matrices, then the forgoing problem was shown to have a *unique* solution, given by eqs.(5.68a–d), which is reproduced below for quick reference:

$$\mathbf{x} = \mathbf{PQb} + \mathbf{Rd} \tag{6.5a}$$

In the foregoing expression, $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{R}$ are the $n \times n$-, $n \times q$- and $n \times p$ matrices that follow:

$$\mathbf{P} = \mathbf{1}_n - \mathbf{RC} \tag{6.5b}$$

$$\mathbf{Q} = (\mathbf{A}^T\mathbf{WA})^{-1}\mathbf{A}^T\mathbf{W} \tag{6.5c}$$

$$\mathbf{R} = (\mathbf{A}^T\mathbf{WA})^{-1}\mathbf{C}^T[\mathbf{C}(\mathbf{A}^T\mathbf{WA})^{-1}\mathbf{C}^T]^{-1} \tag{6.5d}$$

and $\mathbf{1}_n$ is the $n \times n$ identity matrix.

As pointed out in Subsection 5.4.2, the above expression is unsuitable for numerical implementation. A common alternative approach to obtain the solution under study consists in partitioning $\mathbf{C}$ into a $p \times p$ and a $p \times (n-p)$ submatrices, where care should be taken so as to choose a well-conditioned $p \times p$ matrix, for safe inversion. Correspondingly, vector $\mathbf{x}$ should be partitioned into a *master* part $\mathbf{x}_M$, of $n-p$ components, and a *slave* part $\mathbf{x}_S$ of p components. Thus, the constraint equations would be solved for the slave part in terms of the master part and the problem would reduce to an unconstrained least-square problem of dimension $n-p$. However, an arbitrary partitioning of $\mathbf{C}$ may lead to an ill-conditioned $p \times p$ block, even if $\mathbf{C}$ itself is well-conditioned. This situation can be prevented if, out of all N possible partitionings of $\mathbf{C}$, the one with the lowest condition number is chosen. Note that the number of partitionings is given by

$$N = \frac{n!}{p!(n-p)!}$$

and hence, N can become quite large, even for modest values of n and p. Since calculating the condition number of a matrix is a computationally costly procedure, this approach does not seem very attractive.

Alternatively, by introduction of the singular values of $\mathbf{C}$ (Strang, 1988), a subsystem of p equations in p unknowns, which are linear combinations of the components of $\mathbf{x}$, can be found that is optimally conditioned. The computation of singular values, however, similar to that of eigenvalues, is a problem even more difficult to solve than the one at hand, for it is nonlinear and must be solved iteratively. Therefore, it is not advisable to follow the singular-value approach either.

One more approach is followed here, which stems from the geometrical interpretation of the solution (6.5a). Indeed, vector $\mathbf{Qb}$ of that solution represents the unconstrained least-square approximation of eq.(6.2). The second term of the right-hand side of eq.(6.5a) is the minimum-norm solution of the underdetermined system (6.3), based on the norm defined as

$$||\mathbf{x}||_W^2 = \mathbf{x}^T\mathbf{A}^T\mathbf{W}\mathbf{A}\mathbf{x} \tag{6.6}$$

Thus, $\mathbf{P}$ is a *projector*[1] onto the nullspace of $\mathbf{C}$. Indeed, one can readily prove that every n-dimensional vector $\mathbf{x}$ is mapped by $\mathbf{P}$ onto the nullspace of $\mathbf{C}$. Moreover, $\mathbf{P}^2$ can be proven to equal $\mathbf{P}$, thereby making apparent that $\mathbf{P}$ is, in fact, a projector. Furthermore, for any $p \times n$ matrix $\mathbf{C}$, the range of $\mathbf{C}$ and the nullspace of $\mathbf{C}$ are orthogonal subspaces[2] of $\mathbb{R}^n$, their *direct sum* producing all of $\mathbb{R}^n$; i.e., every n-dimensional vector $\mathbf{x}$ can be *uniquely* decomposed into a vector lying in the range of $\mathbf{C}^T$ and a second one lying in the nullspace of $\mathbf{C}$. Now let $\mathbf{L}$ be a $n \times (n-p)$ matrix spanning the nullspace of $\mathbf{C}$, i.e.,

$$\mathbf{CL} = \mathbf{O}_{pn'} \tag{6.7}$$

where $\mathbf{O}_{pn'}$ represents the $p \times (n-p)$ zero matrix. Matrix $\mathbf{L}$ is known as an *orthogonal complement* of matrix $\mathbf{C}$. Thus, the solution to the above problem can be decomposed into two parts, namely,

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{x}_U \tag{6.8}$$

[1]Note that $\mathbf{P}$ is apparently not symmetric!

[2]Note that $\mathcal{R}(\mathbf{C})$ need not be orthogonal to $\mathcal{N}(\mathbf{C})$; in fact, for two subspaces to be orthogonal, they must be embedded in the same space, but $\mathcal{R}(\mathbf{C}) \subseteq \mathbb{R}^p$ and $\mathcal{N}(\mathbf{C}) \subseteq \mathbb{R}^n$, and, in our case, $n \neq p$.

in which $\mathbf{x}_0$ represents the minimum-norm solution to the constraint equation (6.3), i.e., $\mathbf{x}_0$ lies in the range of $\mathbf{C}^T$, while $\mathbf{x}_U$ lies in the nullspace of $\mathbf{C}$. Vector $\mathbf{x}_0$ is computed by means of an orthogonalization method rendering $\mathbf{C}^T$ in upper-triangular form, as discussed in Subsection 5.4.1, while vector $\mathbf{x}_U$ is computed by means of a linear least-square problem. We outline below the computation of $\mathbf{x}_U$.

Let us define a $q \times q$ matrix $\mathbf{V}$ as the Cholesky factor of the given weighting matrix $\mathbf{W}$, i.e.,

$$\mathbf{W} = \mathbf{V}^T\mathbf{V}$$

Moreover, with $\mathbf{x}_0$ known, $\mathbf{x}_U$ is found as the least-square approximation of

$$\mathbf{VAx}_U = \mathbf{V}(\mathbf{b} - \mathbf{Ax}_0) \tag{6.9}$$

subject to the constraints

$$\mathbf{Cx}_U = \mathbf{0} \tag{6.10}$$

Further, let us represent $\mathbf{x}_U$ as the image of a $(n-p)$-dimensional vector under a transformation given by a $n \times (n-p)$ matrix $\mathbf{L}$, namely,

$$\mathbf{x}_U = \mathbf{Lu} \tag{6.11}$$

with $\mathbf{L}$ defined, in turn, as introduced in eq.(6.7). Equation (6.9) thus becomes

$$\mathbf{VALu} = \mathbf{V}(\mathbf{b} - \mathbf{Ax}_0) \tag{6.12}$$

which is an overdetermined system of n linear equations in $n-p$ unknowns. It is thus apparent that $\mathbf{u}$ can be computed as the *unconstrained* least-square solution of eq.(6.12).

However, matrix $\mathbf{L}$, an orthogonal complement of $\mathbf{C}$, *is not unique*. We have thus reached a crucial point in the solution of the constrained linear least-square problem at hand: How to define $\mathbf{L}$. While $\mathbf{L}$ can be defined in infinitely many forms—notice that, once *any* $\mathbf{L}$ has been found, a multiple of it also satisfies eq.(6.7). We define here a distinct $\mathbf{L}$ such that

$$\mathbf{HL} = \begin{bmatrix} \mathbf{O} \\ \mathbf{1} \end{bmatrix} \tag{6.13}$$

where $\mathbf{1}$ is the $(n-p) \times (n-p)$ identity matrix and $\mathbf{O}$ is the $p \times (n-p)$ zero matrix, while $\mathbf{H}$ is defined as the product of Householder reflections rendering $\mathbf{C}^T$ in upper-triangular from—see Subsection 5.4.1. From eq.(6.13), one can obtain matrix $\mathbf{L}$ without any additional computations, for

$$\mathbf{L} = \mathbf{H}^T \begin{bmatrix} \mathbf{O} \\ \mathbf{1} \end{bmatrix} \tag{6.14}$$

whence it is apparent that $\mathbf{L}$ is isotropic, i.e., its condition number is equal to unity. This means that the left Moore-Penrose generalized inverse $\mathbf{L}^I$ of $\mathbf{L}$ can be computed without roundoff-error amplification. In fact, this inverse reduces to $\mathbf{L}^T$, for

$$\mathbf{L}^I = \left([\mathbf{O}^T \quad \mathbf{1}]\mathbf{H}\mathbf{H}^T \begin{bmatrix} \mathbf{O} \\ \mathbf{1} \end{bmatrix} \right)^{-1} \underbrace{[\mathbf{O}^T \quad \mathbf{1}]\mathbf{H}}_{\mathbf{L}^T} = \mathbf{L}^T \tag{6.15}$$

Once $\mathbf{L}$ is known, equation (6.12) can be solved for $\mathbf{u}$ as the least-square approximation of that system. Then, $\mathbf{x}_U$ is calculated from equation (6.11).

As the reader can readily prove, the two components of $\mathbf{x}$, $\mathbf{x}_U$ and $\mathbf{x}_L$, are orthogonal. For this reason, the foregoing procedure is known as the *Orthogonal-Decomposition Algorithm* (ODA).

6.3 Equality-Constrained Nonlinear Least-Square Problems

The solution of nonlinear least-square problems by means of the ODA is now straightforward: The problem consists in finding the least-square error f of an overdetermined system of nonlinear equations, $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{0}$, i.e.,

$$f(\mathbf{x}) = \frac{1}{2}\boldsymbol{\phi}^T\mathbf{W}\boldsymbol{\phi} \quad \rightarrow \quad \min_{\mathbf{x}} \tag{6.16a}$$

subject to the nonlinear constraints

$$\mathbf{h}(\mathbf{x}) = \mathbf{0} \tag{6.16b}$$

where $\boldsymbol{\phi}$ and $\mathbf{x}$ are q- and n-dimensional vectors, respectively, with $q > n$, and $\mathbf{W}$ is a $q \times q$ positive-definite weighting matrix. Moreover, $\mathbf{h}$ is a l-dimensional vector of nonlinear constraints.

The normality condition of the foregoing constrained problem was derived in Ch. 3 in its dual form, eq.(5.73), and recalled below for quick reference:

$$\mathbf{L}^T\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\phi} = \mathbf{0}_{n'} \tag{6.17}$$

with $\mathbf{0}_{n'}$ denoting the $(n-l)$-dimensional zero vector.

The solution of the problem at hand is obtained iteratively: From an initial guess $\mathbf{x}^0$, *not necessarily feasible*, i.e., with $\mathbf{h}(\mathbf{x}^0) \neq \mathbf{0}$, the sequence $\mathbf{x}^1$, $\mathbf{x}^2$, $\ldots$, $\mathbf{x}^k$, $\mathbf{x}^{k+1}$ is generated as

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k \tag{6.18}$$

The increment $\Delta\mathbf{x}^k$ is computed as the solution of an equality-constrained linear least-square problem, namely,

$$\min_{\Delta\mathbf{x}^k} \frac{1}{2}\mathbf{e}^{k^T}\mathbf{W}\mathbf{e}^k \tag{6.19a}$$

subject to

$$\mathbf{J}(\mathbf{x}^k)\Delta\mathbf{x}^k = -\mathbf{h}(\mathbf{x}^k) \tag{6.19b}$$

with $\mathbf{e}^k$ defined as

$$\mathbf{e}^k \equiv -\boldsymbol{\phi}(\mathbf{x}^k) - \boldsymbol{\Phi}(\mathbf{x}^k)\Delta\mathbf{x}^k \tag{6.19c}$$

Now, for compactness, we introduce a few definitions:

$$\mathbf{h}^k \equiv \mathbf{h}(\mathbf{x}^k), \quad \boldsymbol{\phi}^k \equiv \boldsymbol{\phi}(\mathbf{x}^k), \quad \boldsymbol{\Phi}_k \equiv \boldsymbol{\Phi}(\mathbf{x}^k), \quad \mathbf{J}_k \equiv \mathbf{J}(\mathbf{x}^k) \tag{6.20}$$

while $\mathbf{L}_k$ is defined as the isotropic orthogonal complement of $\mathbf{J}_k$ à la eq.(6.13). Moreover, $\boldsymbol{\Phi}_k$ and $\mathbf{J}_k$ will be assumed to be of full rank throughout, the solution $\Delta\mathbf{x}^k$ of problem (6.19a–c) thus being expressed as

$$\Delta\mathbf{x}^k = \Delta\mathbf{v}^k + \mathbf{L}_k\Delta\mathbf{u}^k \tag{6.21}$$

where $\Delta\mathbf{v}^k$ and $\Delta\mathbf{u}^k$ are the minimum-norm and the least-square solutions to an underdetermined and an overdetermined system, namely,

$$\mathbf{J}_k\Delta\mathbf{v}^k = -\mathbf{h}^k \tag{6.22a}$$

$$\mathbf{V}\boldsymbol{\Phi}_k\mathbf{L}_k\Delta\mathbf{u}^k = \mathbf{V}(-\boldsymbol{\phi}^k - \boldsymbol{\Phi}_k\Delta\mathbf{v}^k) \tag{6.22b}$$

The stopping criteria of the procedure are, then,

$$||\Delta\mathbf{x}^k|| \le \epsilon_1 \quad \text{and} \quad ||\mathbf{h}(\mathbf{x}^k)|| \le \epsilon_2 \tag{6.23}$$

for prescribed tolerances ϵ_1 and ϵ_2. These criteria are verified when both the normality condition (6.17) *and the constraint* (6.16b) are verified within the given tolerances. Moreover, we can rewrite eq.(6.22a) (for simplicity, we drop the subscript k) as

$$\mathbf{J}\mathbf{H}^T\mathbf{H}\Delta\mathbf{v} = -\mathbf{h}$$

or

$$(\mathbf{H}\mathbf{J}^T)^T\underbrace{\mathbf{H}\Delta\mathbf{v}}_{\mathbf{w}} = -\mathbf{h} \tag{6.24}$$

Furthermore, note that

$$\mathbf{H}\mathbf{J}^T = \begin{bmatrix} \mathbf{U}_{p\times p} \\ \mathbf{O}_{(n-p)\times p} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} \mathbf{w}_U \\ \mathbf{w}_L \end{bmatrix}$$

where $\mathbf{w}_U$ and $\mathbf{w}_L$ are, correspongingly, p- and $n-p$-dimensional vectors. Hence, eq.(6.24) can be written in the form

$$\begin{bmatrix} \mathbf{U}^T & \mathbf{O}^T \end{bmatrix} \begin{bmatrix} \mathbf{w}_U \\ \mathbf{w}_L \end{bmatrix} = -\mathbf{h}$$

or

$$\mathbf{U}^T\mathbf{w}_U + \mathbf{O}^T\mathbf{w}_L = -\mathbf{h}$$

Last eqaution can be solved for $\mathbf{w}_U$: $\mathbf{w}_U = -\mathbf{U}^{-T}\mathbf{h}$. Whence, from $\mathbf{H}\mathbf{v} \equiv \mathbf{w}$ we can obtain $\mathbf{v} = \mathbf{H}^T\mathbf{w}$. Now, from eq.(6.22b) $\Delta\mathbf{x}^k$ can be expressed as

$$\Delta\mathbf{x}^k = (\mathbf{M}\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\Phi} - \mathbf{1}_n)\mathbf{J}^\dagger\mathbf{h} - \mathbf{M}\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\phi} \tag{6.25}$$

where subscripts and superscripts have been dropped from the right-hand side for compactness, $\mathbf{1}_n$ represents the $n \times n$ identity matrix, $\mathbf{J}^\dagger$ is the right Moore-Penrose generalized inverse of $\mathbf{J}$, and $\mathbf{M}$ is the $n \times n$ matrix defined as

$$\mathbf{M} = \mathbf{L}(\mathbf{L}^T\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\Phi}\mathbf{L})^{-1}\mathbf{L}^T \tag{6.26}$$

Upon convergence, the constraint equations hold. Therefore, $\mathbf{h} = \mathbf{0}$ and the normality condition (6.17) holds. Therefore, upon convergence, $\Delta\mathbf{x}^k \to \mathbf{0}$.

The sequence $\{\Delta\mathbf{x}^k\}$ produces a sequence $\{f_k\}$, the increment Δf between two consecutive values of the sequence being given by

$$\Delta f = (\nabla f)^T\Delta\mathbf{x} \tag{6.27}$$

where ∇f is the gradient of f, i.e., $\nabla f = \boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\phi}$, and hence,

$$\begin{aligned} \Delta f &= (\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\phi})^T\Delta\mathbf{x} \\ &= -\boldsymbol{\phi}^T\mathbf{W}\boldsymbol{\Phi}\mathbf{M}\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\phi} - \boldsymbol{\phi}^T\mathbf{W}\boldsymbol{\Phi}(\mathbf{1} - \mathbf{M}\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\Phi})\mathbf{J}^\dagger\mathbf{h} \end{aligned} \tag{6.28}$$

From eq.(6.28), if the current value of $\mathbf{x}$ is feasible, i.e., if $\mathbf{h} = \mathbf{0}$, then Δf is negative-definite, and the procedure yields an improved value of f. On the other hand, one can readily verify that

$$\Delta(\mathbf{h}^T\mathbf{h}) \equiv \mathbf{h}_{k+1}^T\mathbf{h}_{k+1} - \mathbf{h}_k^T\mathbf{h}_k = -2\mathbf{h}^T\mathbf{h} = 0 \tag{6.29}$$

which is negative definite. Therefore, the procedure gives a sequence of $\mathbf{x}$ values that approaches the constraints.

Example 6.3.1 (A Quadratic Objective Function with a Quadratic Constraint)

We recall Example 5.6.1, which is reproduced below for quick reference:

$$f(\mathbf{x}) = \frac{1}{2}(9x_1^2 - 8x_1x_2 + 3x_2^2) \quad \rightarrow \quad \min_{x_1, x_2}$$

subject to

$$h(\mathbf{x}) = x_1^2 + x_2^2 - 1 = 0$$

While the objective function is quadratic in a linear function of the design-variable vector, the constraint is nonlinear, which disqualifies this problem from a direct solution, as found in Section 6.2 for linear least-squares subject to linear constraints. This problem, due to its simplicity, could be solved exactly in Chapter 3. Here, we solve this problem numerically, using the ODA. First, note that the objective function $f(\mathbf{x})$ *can be factored as*

$$f(\mathbf{x}) = \frac{1}{2}\boldsymbol{\phi}^T\mathbf{W}\boldsymbol{\phi}$$

with

$$\mathbf{W} = \begin{bmatrix} 9 & -4 \\ -4 & 3 \end{bmatrix} \quad \text{and} \quad \boldsymbol{\phi} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

i.e, $f(\mathbf{x})$ *is a special case of the* $f(\mathbf{x})$ *defined in eq.(6.2), with* $\mathbf{A} = \mathbf{1}$ *and* $\mathbf{b} = \mathbf{0}$. *We include below a Maple worksheet describing the step-by-step implementation of the ODA in solving the foregoing problem* iteratively.

```
> restart:with(linalg):

Warning, the protected names norm and trace have been redefined and
unprotected

> with(plots): with(plottools):

Warning, the name changecoords has been redefined

Warning, the name arrow has been redefined
```

Linear-least square problem subject to a quadratic constraint

$$f(x) = (1/2)(9x_1^2 - 8x_1x_2 + 3x_2^2) \quad \rightarrow \quad \min_{x_1, x_2}$$

subject to

$$h(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0$$

```
> obj:= proc(x) (1/2)*(9*x[1]^2 - 8*x[1]*x[2] +
> 3*x[2]^2)
> end; #procedure to compute the objective function
```

$$obj := \mathbf{proc}(x)\, 9/2 * x_1{}^2 - 4 * x_1 * x_2 + 3/2 * x_2{}^2\, \mathbf{end\ proc}$$

```
> constr:= proc(x) x[1]^2+x[2]^2 - 1 end;
> #procedure computing the constraint
```

$$constr := \mathbf{proc}(x)\, x_1{}^2 + x_2{}^2 - 1\, \mathbf{end\ proc}$$

```
> dhdx:= proc(x) matrix([[2*x[1], 2*x[2]]])
> end; #procedure computing the gradient of the constraint
```

$$dhdx := \mathbf{proc}(x)\, \mathrm{matrix}([[2 * x_1,\ 2 * x_2]])\, \mathbf{end\ proc}$$

```
> alfa:= proc(J)
> evalf(signum(J[1,1])*sqrt(J[1,1]^2 + J[1,2]^2))
> end; #procedure computing "alpha" of Householder reflections in
> least-square solution at each iteration
```

$$alfa := \mathbf{proc}(J)\, \mathrm{evalf}(\mathrm{signum}(J_{1,1}) * \mathrm{sqrt}(J_{1,1}{}^2 + J_{1,2}{}^2))\, \mathbf{end\ proc}$$

```
> W:=matrix([[9, -4], [-4, 3]]); #weighting matrix
```

$$\mathrm{W} := \begin{bmatrix} 9 & -4 \\ -4 & 3 \end{bmatrix}$$

```
> V:=transpose(cholesky(W)); #Maple returns a
> lower-triangular matrix with procedure "cholesky"!
```

$$V := \begin{bmatrix} 3 & \dfrac{-4}{3} \\ 0 & \dfrac{1}{3}\sqrt{11} \end{bmatrix}$$

```
> V:= map(evalf, V);
```

$$V := \begin{bmatrix} 3. & -1.333333333 \\ 0. & 1.105541597 \end{bmatrix}$$

```
> ID:=Matrix(2,2,shape=identity);
> E:= matrix([[0], [1]]);
> Phi:= ID;
> B:=evalm(V&*Phi); #Defining various auxiliary matrices
```

$$ID := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$E := \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\Phi := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$B := \begin{bmatrix} 3. & -1.333333333 \\ 0. & 1.105541597 \end{bmatrix}$$

```
> x:=vector([2, 2]); x0:= evalm(x); #initial
> guess, x^0, stored as x0  for plotting
```

$$x := [2, 2]$$

$$x0 := [2, 2]$$

```
> f:= evalf(obj(x)); #f_0
```

$$f := 8.$$

```
> phi:= evalm(x); #phi^0
```

$$\phi := [2, 2]$$

```
> h:= constr(x); #h^0
```

$$h := 7$$

```
> J:= dhdx(x); #J_0
```

$$J := \begin{bmatrix} 4 & 4 \end{bmatrix}$$

```
> alpha:= alfa(J); #local variable
```

$$\alpha := 5.656854248$$

```
> t:=vector([J[1,1] + alpha, J[1,2]]);
> #u in HHR algorithm, a local variable
```

$$t := [9.656854248, 4]$$

```
> normt2:=evalf(dotprod(t,t)/2); #half of
> Euclidean norm-squared of t, a local variable
```

$$normt2 := 54.62741700$$

```
> H:=evalm(ID - t&*transpose(t)/normt2);
> #evaluating Householder reflection
```

$$H := \begin{bmatrix} -.707106781 & -.7071067812 \\ -.7071067812 & .7071067811 \end{bmatrix}$$

```
> P:=evalm(H&*transpose(H)); #checking whether H
> is a reflection
```

$$P := \begin{bmatrix} .9999999997 & 0. \\ 0. & .9999999999 \end{bmatrix}$$

```
> detH:=det(H);
```

$$detH := -.9999999998$$

H is indeed a reflection!

```
> HJT:=evalm(H&*transpose(J));
```

$$HJT := \begin{bmatrix} -5.656854249 \\ -.1\,10^{-8} \end{bmatrix}$$

```
> HJT[2,1]:=0; print(HJT); #setting last entry
> of HJ^T equal to zero
```

$$HJT_{2,1} := 0$$

$$\begin{bmatrix} -5.656854249 \\ 0 \end{bmatrix}$$

```
> w:= vector([-h/HJT[1,1], 0]); #w = Hv
```

$$w := [1.237436867, 0]$$

```
> v:=evalm(H&*w); #v^0
```

$$v := [-.8749999997, -.8750000000]$$

```
> L:= evalm(H&*E);
> BL:=evalm(B&*L); #L_0 & (BL)_0
```

$$L := \begin{bmatrix} -.7071067812 \\ .7071067811 \end{bmatrix}$$

$$BL := \begin{bmatrix} -3.064129385 \\ .7817359600 \end{bmatrix}$$

```
> p:=matadd(phi, Phi&*v); #auxiliary variable
```

$$p := [1.125000000, 1.125000000]$$

```
> r:= evalm(-V&*p);
> #RHS of overdetermined system to compute u in ODA
```

$$r := [-1.875000000, -1.243734297]$$

```
> u:= leastsqrs(BL, r); #u^0
```

$$u := [.4772970772]$$

```
> Deltax:= matadd(v, L&*u); #Deltax^0
```

$$Deltax := [-1.212500000, -.5375000001]$$

First iteration is complete. Update **x**:

```
> x:= evalm(x + Deltax); x1:= evalm(x); #x^1
```

$$x := [.787500000, 1.462500000]$$

$$x1 := [.787500000, 1.462500000]$$

```
> f:= evalf(obj(x)); #f_1
```

$$f := 1.392187500$$

```
> phi:= evalm(x); #phi^1
```

$$\phi := [.787500000, 1.462500000]$$

```
> h:= constr(x); #h^1
```

$$h := 1.759062500$$

```
> J:= dhdx(x); #J_1
```

$$J := \begin{bmatrix} 1.575000000 & 2.925000000 \end{bmatrix}$$

```
> alpha:= alfa(J);
```

$$\alpha := 3.322085189$$

```
> t:=vector([J[1,1] + alpha, J[1,2]]);
> #u in HHR algorithm
```

$$t := [4.897085189, 2.925000000]$$

```
> normt2:=evalf(dotprod(t,t)/2);
```

$$normt2 := 16.26853418$$

```
> H:=evalm(ID - t&*transpose(t)/normt2);
> #evaluating Householder reflection
```

$$H := \begin{bmatrix} -.474099823 & -.8804710997 \\ -.8804710997 & .4740998233 \end{bmatrix}$$

```
> HJT:=evalm(H&*transpose(J));
```

$$HJT := \begin{bmatrix} -3.322085188 \\ .1\,10^{-8} \end{bmatrix}$$

```
> HJT[2,1]:=0; print(HJT);
```

$$HJT_{2,1} := 0$$

$$\begin{bmatrix} -3.322085188 \\ 0 \end{bmatrix}$$

```
> w:= vector([-h/HJT[1,1], 0]); #w = Hv
```

$$w := [.5295055366, 0]$$

```
> v:=evalm(H&*w); # v^1
```

$$v := [-.2510384812, -.4662143221]$$

```
> L:= evalm(H&*E); BL:=evalm(V&*Phi&*L);
> #L_1 & (BL)_1
```

$$L := \begin{bmatrix} -.8804710997 \\ .4740998233 \end{bmatrix}$$

$$BL := \begin{bmatrix} -3.273546397 \\ .5241370758 \end{bmatrix}$$

```
> p:=matadd(phi, Phi&*v); #auxiliary variable
```

$$p := [.5364615188, .9962856779]$$

```
> r:= evalm(-V&*p);
> #RHS of overdetermined system to compute u in ODA
```

$$r := [-.281003652, -1.101435259]$$

```
> u:= leastsqrs(BL, r); #u^1
```

$$u := [.03116921755]$$

```
> Deltax:= matadd(v, L&*u); #delta x^1
```

$$Deltax := [-.2784820764, -.4514370016]$$

Second iteration is complete. Update $\mathbf{x}$:

```
> x:= matadd(x, Deltax); x2:= evalm(x); #x^2
```

$$x := [.5090179236, 1.011062998]$$

$$x2 := [.5090179236, 1.011062998]$$

```
> f:= obj(x); #f_2
```

$$f := .640722436$$

```
> phi:= evalm(x); #phi^2
```

$$\phi := [.5090179236, 1.011062998]$$

```
> h:= constr(x); #h^2
```

$$h := .281347632$$

```
> J:= dhdx(x); #J_2
```

$$J := \begin{bmatrix} 1.018035847 & 2.022125996 \end{bmatrix}$$

```
> alpha:= alfa(J);
```

$$\alpha := 2.263932537$$

```
> t:=vector([J[1,1] + alpha, J[1,2]]);
> #u in HHR algorithm
```

$$t := [3.281968384, 2.022125996]$$

```
> normt2:=evalf(dotprod(t,t)/2);
```

$$normt2 := 7.430155005$$

```
> H:=evalm(ID - t&*transpose(t)/normt2);
> #evaluating Householder
> reflection
```

$$H := \begin{bmatrix} -.449675877 & -.8931918088 \\ -.8931918088 & .4496758759 \end{bmatrix}$$

```
> HJT:=evalm(H&*transpose(J));
```

$$HJT := \begin{bmatrix} -2.263932538 \\ -.12\,10^{-8} \end{bmatrix}$$

```
> HJT[2,1]:=0; print(HJT);
```

$$HJT_{2,1} := 0$$

$$\begin{bmatrix} -2.263932538 \\ 0 \end{bmatrix}$$

```
> w:= vector([-h/HJT[1,1], 0]); #w = Hv
```

$$w := [.1242738586, 0]$$

```
> v:=evalm(H&*w); #v^2
```

$$v := [-.05588295635, -.1110003925]$$

```
> L:= evalm(H&*E); BL:=evalm(V&*Phi&*L);
> #L_2 & (BL)_2
```

$$L := \begin{bmatrix} -.8931918088 \\ .4496758759 \end{bmatrix}$$

$$BL := \begin{bmatrix} -3.279143260 \\ .4971353860 \end{bmatrix}$$

```
> p:=matadd(phi, Phi&*v); #auxiliary variable
```

$$p := [.4531349672, .9000626055]$$

```
> r:= evalm(-V&*p);
> #RHS of overdetermined system to compute u in ODA
```

$$r := [-.159321428, -.9950566503]$$

```
> u:= leastsqrs(BL, r); #u^2
```

$$u := [.002523646038]$$

```
> Deltax:= matadd(v, L&*u); #Deltax^2
```

$$Deltax := [-.05813705632, -.1098655698]$$

Third iteration is complete. Update $\mathbf{x}$:

```
> x:= matadd(x, Deltax); x3:=evalm(x); #x^3
```

$$x := [.4508808673, .9011974282]$$

$$x3 := [.4508808673, .9011974282]$$

```
> f:= obj(x); #f_3
```

$$f := .5077254992$$

```
> phi:= evalm(x); #phi^3
```

$$\phi := [.4508808673, .9011974282]$$

```
> h:= constr(x); #h^3
```

$$h := .015450361$$

```
> J:= dhdx(x); #J_3
```

$$J := \begin{bmatrix} .9017617346 & 1.802394856 \end{bmatrix}$$

```
> alpha:= alfa(J);
```

$$\alpha := 2.015391139$$

```
> t:=vector([J[1,1] + alpha, J[1,2]]);
> #u in HHR algorithm
```

$$t := [2.917152874, 1.802394856]$$

```
> normt2:=evalf(dotprod(t,t)/2);
```

$$normt2 := 5.879204055$$

```
> H:=evalm(ID - t&*transpose(t)/normt2);
> #evaluating Householder reflection
```

$$H := \begin{bmatrix} -.447437580 & -.8943151635 \\ -.8943151635 & .4474375804 \end{bmatrix}$$

```
> HJT:=evalm(H&*transpose(J));
```

$$HJT := \begin{bmatrix} -2.015391138 \\ .2\,10^{-9} \end{bmatrix}$$

```
> HJT[2,1]:=0; print(HJT);
```

$$HJT_{2,1} := 0$$

$$\begin{bmatrix} -2.015391138 \\ 0 \end{bmatrix}$$

```
> w:= vector([-h/HJT[1,1], 0]); #w = Hv
```

$$w := [.007666184846, 0]$$

```
> v:=evalm(H&*w); #v^3
```

$$v := [-.003430139195, -.006855985354]$$

```
> L:= evalm(H&*E); BL:=evalm(V&*Phi&*L);
> L_3 & (BL)_3
```

$$L := \begin{bmatrix} -.8943151635 \\ .4474375804 \end{bmatrix}$$

$$BL := \begin{bmatrix} -3.279528930 \\ .4946608572 \end{bmatrix}$$

```
> p:=matadd(phi, Phi&*v); #auxiliary variable
```

$$p := [.4474507281, .8943414428]$$

```
> r:= evalm(-V&*p);
> #RHS of overdetermined system to compute u in ODA
```

$$r := [-.149896927, -.9887316669]$$

```
> u:= leastsqrs(BL, r); #u^3
```

$$u := [.0002276777133]$$

```
> Deltax:= matadd(v, L&*u); #Deltax^3
```

$$Deltax := [-.003633754826, -.006754113789]$$

Fourth iteration is complete. Update **x**:

```
> x:= matadd(x, Deltax); x4:=evalm(x); # x^4
```

$$x := [.4472471125, .8944433144]$$

$$x4 := [.4472471125, .8944433144]$$

```
> f:= obj(x); #f_4
```

$$f := .5000294132$$

```
> phi:= evalm(x); #phi^4
```

$$\phi := [.4472471125, .8944433144]$$

```
> h:= constr(x); #h^4
```

$$h := .000058822$$

```
> J:= dhdx(x); #J_4
```

$$J := \begin{bmatrix} .8944942250 & 1.788886629 \end{bmatrix}$$

```
> alpha:= alfa(J);
```

$$\alpha := 2.000058822$$

```
> t:=vector([J[1,1] + alpha, J[1,2]]);
> #u in HHR algorithm
```

$$t := [2.894553047, 1.788886629]$$

```
> normt2:=evalf(dotprod(t,t)/2);
```

$$normt2 := 5.789276355$$

```
> H:=evalm(ID - t&*transpose(t)/normt2);
> #evaluating Householder
> reflection
```

$$H := \begin{bmatrix} -.447233960 & -.8944170093 \\ -.8944170093 & .4472339590 \end{bmatrix}$$

```
> HJT:=evalm(H&*transpose(J));
```

$$HJT := \begin{bmatrix} -2.000058823 \\ -.3\,10^{-9} \end{bmatrix}$$

```
> HJT[2,1]:=0; print(HJT);
```

$$HJT_{2,1} := 0$$

$$\begin{bmatrix} -2.000058823 \\ 0 \end{bmatrix}$$

```
> w:= vector([-h/HJT[1,1], 0]); #w = Hv
```

$$w := [.00002941013500, 0]$$

```
> v:=evalm(H&*w); #v^4
```

$$v := [-.00001315321114, -.00002630492499]$$

```
> L:= evalm(H&*E); BL:=evalm(V&*Phi&*L);
> #L_4 & (BL)_4
```

$$L := \begin{bmatrix} -.8944170093 \\ .4472339590 \end{bmatrix}$$

$$BL := \begin{bmatrix} -3.279562973 \\ .4944357453 \end{bmatrix}$$

```
> p:=matadd(phi, Phi&*v); #auxiliary variable
```

$$p := [.4472339593, .8944170095]$$

```
> r:= evalm(-V&*p);
> #RHS of overdetermined system to compute u in ODA
```

$$r := [-.149145866, -.9888152091]$$

```
> u:= leastsqrs(BL, r); #u^4
```

$$u := [.00002069770909]$$

```
> Deltax:= matadd(v, L&*u); #Deltax^4
```

$$Deltax := [-.00003166559420, -.00001704820661]$$

Fourth iteration is complete. Update **x**:

```
> x:= matadd(x, Deltax);  x4:= evalm(x); #x^4
```

$$x := [.4472154469, .8944262662]$$

$$x4 := [.4472154469, .8944262662]$$

```
> f:= obj(x); #f_4
```

$$f := .5000000006$$

Given the norm of Deltax, we declare convergence here, and plot the iteration history in $x[1]$-$x[2]$ plane:

```
> o0:= evalm(x0); o1:= evalm(x1);
> o2:= evalm(x2);  o3:= evalm(x3); o4:= evalm(x4);
```

$$o0 := [2, 2]$$

$$o1 := [.787500000, 1.462500000]$$

$$o2 := [.5090179236, 1.011062998]$$

$$o3 := [.4508808673, .9011974282]$$

$$o4 := [.4472154469, .8944262662]$$

```
> p0:=point(convert(o0,list), symbol=circle,
> color=blue);
> p1:=point(convert(o1,list), symbol=circle, color=blue);
> p2:=point(convert(o2,list), symbol=circle, color=blue);
> p3:=point(convert(o3,list), symbol=circle, color=blue);
> p4:=point(convert(o4,list), symbol=circle, color=blue);
```

p0 := POINTS([2., 2.], COLOUR(*RGB*, 0., 0., 1.00000000), SYMBOL(*CIRCLE*))

p1 := POINTS([.787500000, 1.462500000], COLOUR(*RGB*, 0., 0., 1.00000000), SYMBOL(*CIRCLE*))

p2 := POINTS([.5090179236, 1.011062998], COLOUR(*RGB*, 0., 0., 1.00000000), SYMBOL(*CIRCLE*))

p3 := POINTS([.4508808673, .9011974282], COLOUR(*RGB*, 0., 0., 1.00000000), SYMBOL(*CIRCLE*))

p4 := POINTS([.4472154469, .8944262662], COLOUR(*RGB*, 0., 0., 1.00000000), SYMBOL(*CIRCLE*))

```
> l1 :=
> arrow(convert(o0,list),convert(o1,list), 10.0, 0.1, .1, arrow,
> color=red, thickness=2):
> l2 := arrow(convert(o1,list),convert(o2,list), 6.0, 0.1, .2, arrow,
> color=green, thickness=2):
> l3 := arrow(convert(o2,list),convert(o3,list), 6.0, 0.1, .7, arrow,
> color=red, thickness=2):
> l4 := arrow(convert(o3,list),convert(o4,list), 6.0, 0.1, 6.0, arrow,
> color=green, thickness=2):
> c1 := arc([0,0], 1,-Pi/6..4*Pi/6,color=black, thickness=2):
> obj_plot:= proc(ax,ay) (1/2)*(9*ax^2 - 8*ax*ay + 3*ay^2)
> end:f1:=implicitplot(obj_plot-3,-0.5..2,-0.5..2.5,numpoints=3000,
> linestyle=4,color=blue):
> f2:=implicitplot(obj_plot-2,-0.5..2,-0.5..2,
> numpoints=3000,linestyle=4,color=blue):
> f3:=implicitplot(obj_plot-1,-0.5..2,-0.5..2,numpoints=6000,
> linestyle=4,color=blue):f4:=implicitplot(obj_plot-0.5,-0.5..2,-0.5..2,
> numpoints=6000, linestyle=4,color=blue):

> display({c1,p0,p1, p2, p3, p4, p1, l1, l2,
> l3, l4,f1,f2,f3,f4},
> insequence = false, color=red, scaling=constrained);
```

The plots produced by the plotting commands in the Maple worksheet are reproduced in Fig. 6.1. A plot of the contours of the objective function and the constraint, showing all four stationary points, is displayed in Fig. 6.2.

Example 6.3.2 (Finding the Eigenvalues and Eigenvectors of a Symmetric Matrix)

The problem of finding the eigenvalues and corresponding eigenvectors of a $n \times n$ symmetric positive-definite matrix $\mathbf{M}$ is solved as a linear least-square problem subject to quadratic constraints: For $i = 1, 2, \cdots, n$, and $k = 1, 2, \cdots, i$, find λ_i and $\mathbf{x}_i$ such that

$$\lambda_i = \min_{\mathbf{x}_i} \frac{1}{2}\mathbf{x}_i^T \mathbf{M}\mathbf{x}_i$$

subject to

$$\mathbf{x}_k^T \mathbf{x}_i = \begin{cases} 0, & \text{if } k = 1,\ 2,\ \cdots,\ i-1; \\ 1, & \text{if } k = i. \end{cases}$$

where λ_i is the ith eigenvalue of matrix $\mathbf{M}$ and $\mathbf{x}_i$ is the corresponding eigenvector.

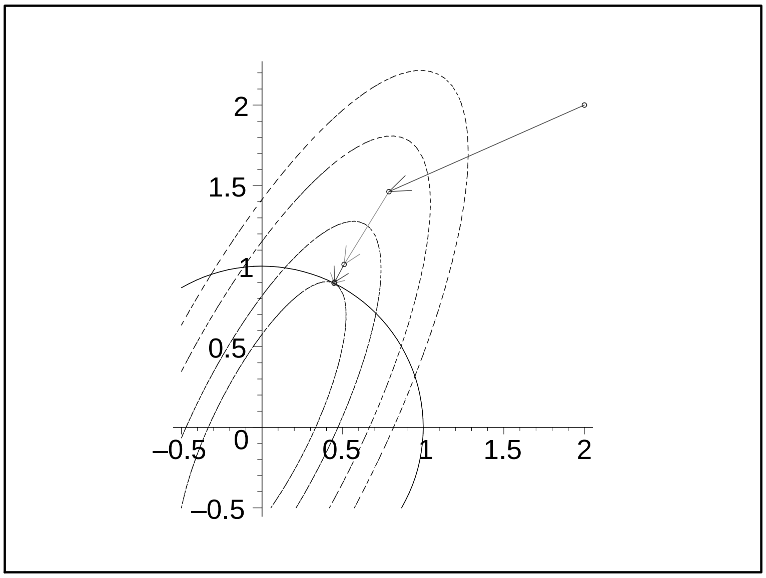


Figure 6.1: The four iterations leading to the solution of the linear least-square problem subject to one quadratic constraint

In order to use the ODA package to solve the problem, we define, for $i = 1, 2, \cdots, n$*:*

$$\begin{aligned} q &= n \qquad \text{and} \qquad l = i, \\ \mathbf{x} &= \mathbf{x}_i, \\ \boldsymbol{\phi}(\mathbf{x}) &= \mathbf{x}, \\ \mathbf{h}(\mathbf{x}) &= [\, \mathbf{x}_1^T\mathbf{x} \quad \cdots \quad \mathbf{x}_{i-1}^T\mathbf{x} \quad \mathbf{x}^T\mathbf{x} - 1 \,]^T, \\ \mathbf{W} &= \mathbf{M} \end{aligned} \tag{6.30}$$

where $\mathbf{x}_k$*, for* $k = 1, 2, \cdots, i-1$*, are the previously calculated eigenvectors of* $\mathbf{M}$*, and hence, are known. With the above definitions, for* $i = 1, 2, \cdots, n$*, subroutine* `LSSCNL` *of the ODA package is called* n *times. After each call, one eigenvalue and its corresponding eigenvector are obtained. Notice that, in the last call, the number of constraints is equal to the number of variables, namely,* $l = n$*. Matrix* $\mathbf{M}$ *is given as*

$$\mathbf{M} = \begin{bmatrix} 4 & 2 & 1 & 1 & 1 \\ 2 & 4 & 2 & 1 & 1 \\ 1 & 2 & 4 & 2 & 1 \\ 1 & 1 & 2 & 4 & 2 \\ 1 & 1 & 1 & 2 & 4 \end{bmatrix}.$$

We use the initial guess

$$\mathbf{x}^0 = [\, 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \,]^T.$$

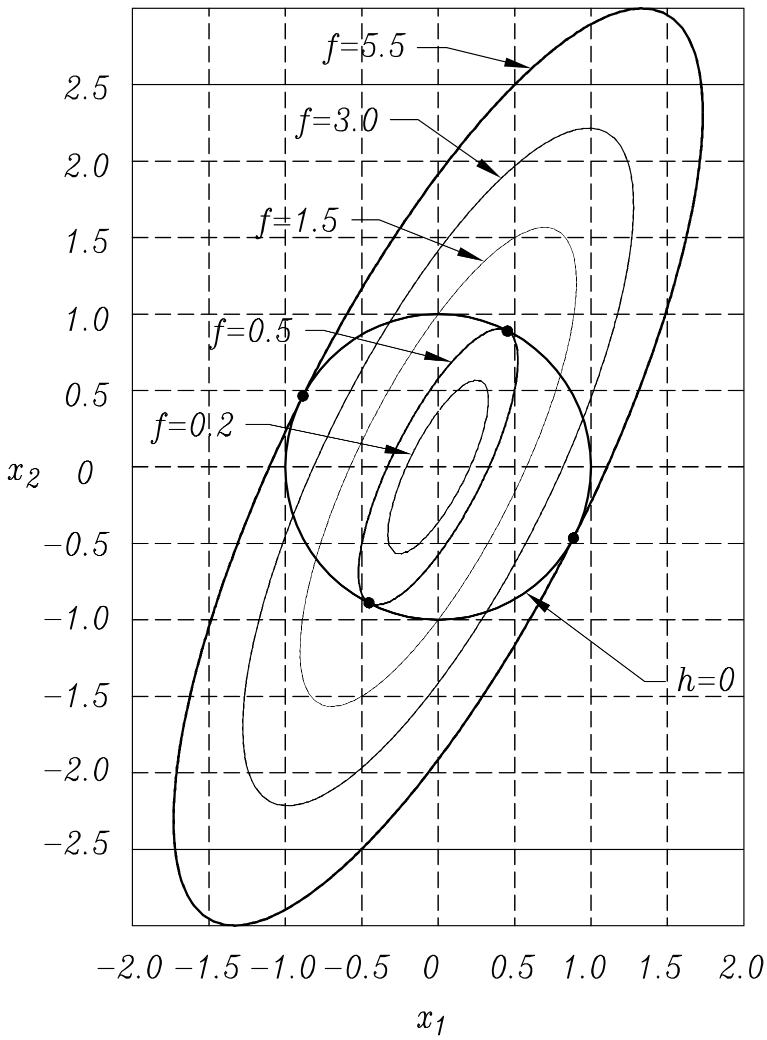


Figure 6.2: The contours of constant f and the constraint $h = 0$

The eigenvalues and the eigenvectors computed with the ODA package are listed in Table 6.1. In that table, the number of iterations that the package took till convergence was reached with $\epsilon_1 = 0.0001$ *and* $\epsilon_2 = 0.0001$*, is indicated.*

6.4 Equality-Constrained Optimization with Arbitrary Objective Function

The problem to be solved is defined as:

$$f = f(\mathbf{x}) \quad \rightarrow \quad \min_{\mathbf{x}} \tag{6.31}$$

i	1	2	3	4	5
λ_i	1.27738	3.08749	9.63513	2.0	4.0
1st comp. of $\mathbf{x}_i$	0.26565	-0.51369	0.40689	0.5	0.5
2nd comp. of $\mathbf{x}_i$	-0.51853	0.10368	0.46944	-0.5	0.5
3rd comp. of $\mathbf{x}_i$	0.56667	0.67138	0.47764	0.0	0.0
4th comp. of $\mathbf{x}_i$	-0.51853	0.10368	0.46944	0.5	-0.5
5th comp. of $\mathbf{x}_i$	0.26565	-0.51369	0.40689	-0.5	-0.5
# of iterations	12	11	6	38	54

Table 6.1: Eigenvalues and eigenvectors of $\mathbf{M}$

subject to the nonlinear equality constraints

$$\mathbf{h}(\mathbf{x}) = \mathbf{0} \tag{6.32}$$

where $\mathbf{x}$ is the n-dimensional design-variable vector, the objective function $f(\mathbf{x})$ being a nonlinear function of $\mathbf{x}$, not necessarily quadratic in the sense of Section 6.3, but with continuous derivatives up to the second order. Moreover, $\mathbf{h}(\mathbf{x})$ is a l-dimensional vector of nonlinear equality constraints, with a continuous gradient.

In the problem defined in eq.(6.31), if the constraints in eq.(6.32) are *analytic*, then there exists a *feasible manifold* $\mathcal{F} \subset \mathbb{R}^n$, of dimension l, such that, if $\mathbf{u} \in \mathcal{F}$, then

$$\mathbf{h}(\mathbf{x}(\mathbf{u})) = \mathbf{0} \tag{6.33}$$

In the particular case in which $\mathbf{h}(\mathbf{x}(\mathbf{u}))$ is linear, then $\mathcal{F}$ is a vector space, i.e., the *feasible space* of the problem at hand.

At $\mathbf{x} = \mathbf{x}^k$, we assume that, in general, $\mathbf{h}(\mathbf{x}^k) = \mathbf{h}^k \neq \mathbf{0}$, i.e., the current $\mathbf{x}$ is not feasible, and note that $f(\mathbf{x}^k + \Delta\mathbf{x}^k)$ can be expanded, to a second order, as

$$f(\mathbf{x}^k + \Delta\mathbf{x}^k) \approx f(\mathbf{x}^k) + (\nabla f)_k^T \Delta\mathbf{x}^k + \frac{1}{2}(\Delta\mathbf{x}^k)^T(\nabla\nabla f)_k \Delta\mathbf{x}^k \rightarrow \min_{\Delta\mathbf{x}^k} \tag{6.34a}$$

subject to

$$\mathbf{J}_k \Delta\mathbf{x}^k = -\mathbf{h}^k \tag{6.34b}$$

We have thus derived a linear least-square problem in $\Delta\mathbf{x}^k$ subject to the linear constraints (6.34b). To find the increment $\Delta\mathbf{x}^k$, we resort to the ODA, as introduced in Section 6.2. To this end, we decompose the foregoing vector into its two orthogonal components:

$$\Delta\mathbf{x}^k = \Delta\mathbf{x}_0^k + \mathbf{L}_k \Delta\mathbf{u}^k$$

where

$$\Delta \mathbf{x}_0^k = -\mathbf{J}_k^T(\mathbf{J}_k\mathbf{J}_k^T)^{-1}\mathbf{h}^k \tag{6.35}$$

is the minimum-norm solution of eq.(6.34b), and $\mathbf{L}_k$ is the isotropic orthogonal complement of $\mathbf{J}_k$ defined in eqs.(6.13) and (6.14), while $\mathbf{J}_k$ itself is defined as the gradient of $\mathbf{h}$ with respect to $\mathbf{x}$, evaluated at $\mathbf{x} = \mathbf{x}^k$. Moreover, $\mathbf{L}_k$ and $\Delta\mathbf{u}^k$ are found with the procedure described in Section 6.2 for linearly-constrained linear least-square problems. Furthermore, with $\Delta\mathbf{x}_0^k$ given by eq.(6.35), $f(\mathbf{x}^k + \Delta\mathbf{x}^k)$ becomes a function solely of $\Delta\mathbf{u}^k$, i.e.,

$$\begin{aligned} f(\Delta\mathbf{u}^k) \approx \tilde{f}(\Delta\mathbf{u}^k) \equiv f(\mathbf{x}^k) &+ (\nabla f)_k^T(\Delta\mathbf{x}_0^k + \mathbf{L}_k\Delta\mathbf{u}^k) \\ &+\frac{1}{2}(\Delta\mathbf{x}_0^k + \mathbf{L}_k\Delta\mathbf{u}^k)^T(\nabla\nabla f)_k(\Delta\mathbf{x}_0^k + \mathbf{L}_k\Delta\mathbf{u}^k) \rightarrow \min_{\Delta\mathbf{u}^k} \end{aligned}$$

which can be cast in the form

$$\begin{aligned} \tilde{f}(\Delta\mathbf{u}^k) = \frac{1}{2}(\Delta\mathbf{u}^k)^T\mathbf{L}_k^T(\nabla\nabla f)_k\mathbf{L}_k\Delta\mathbf{u}^k &+ \left[\mathbf{L}_k^T(\nabla\nabla f)_k\Delta\mathbf{x}_0^k + \mathbf{L}_k^T(\nabla f)_k\right]^T \Delta\mathbf{u}^k \\ +\frac{1}{2}(\Delta\mathbf{x}_0^k)^T(\nabla\nabla f)_k\Delta\mathbf{x}_0^k &+ (\nabla f)_0^T\Delta\mathbf{x}_0^k + f(\mathbf{x}^k) \rightarrow \min_{\Delta\mathbf{u}^k} \end{aligned} \tag{6.36}$$

subject to *no constraints*, $\tilde{f}(\Delta\mathbf{u}^k)$ being *quadratic* in $\Delta\mathbf{u}^k$. Function $\tilde{f}(\Delta\mathbf{u}^k)$ has a minimum if its Hessian with respect to $\Delta\mathbf{u}^k$, feasible Hessian $\mathbf{H}_k = \mathbf{L}_k^T(\nabla\nabla f)_k\mathbf{L}_k$, is positive-definite. Under the assumption that this is the case, then, the minimum $\Delta\mathbf{u}^k$ of $\tilde{f}(\Delta\mathbf{u}^k)$ can be readily computed upon zeroing its gradient with respect to $\Delta\mathbf{u}^k$, which yields

$$\mathbf{H}_k\Delta\mathbf{u}^k = -\mathbf{L}_k^T(\nabla\nabla f)_k\Delta\mathbf{x}_0^k - \mathbf{L}_k^T(\nabla f)_k$$

Under the assumption that $\mathbf{H}_k$ is positive-definite, it is invertible, and hence,

$$\Delta\mathbf{u}^k = -\mathbf{H}_k^{-1}[\mathbf{L}_k^T(\nabla\nabla f)_k\Delta\mathbf{x}_0^k + \mathbf{L}_k^T(\nabla f)_k] \tag{6.37}$$

We have thus reduced the original problem to a *sequence* of *linear-quadratic programs*. This means that we have solved the problem iteratively. At each iteration, moreover, we find the correction to the current approximation $\Delta\mathbf{x}^k$ by means of a combination of two linear problems, one being a minimum-norm problem, the other involving a determined linear system of equations. For this reason, the above procedure is called *sequential quadratic programming*.

The foregoing procedure relies on the rather daring assumption that the Hessian $\mathbf{H}_k$ is positive-definite. Below we study the more realistic case of a non-positive-definite Hessian.

6.4.1 Sequential Quadratic Programming with Hessian Stabilization

In the presence of a non-positive-definite Hessian $\mathbf{H}_k$, we aim at a *perturbation* $\Delta\mathbf{H}_k$ of the Hessian that will render the *perturbed Hessian* $\tilde{\mathbf{H}}_k$ positive-definite, thus producing

$$\tilde{\mathbf{H}}_k \equiv \mathbf{H}_k + \Delta\mathbf{H}_k \tag{6.38}$$

How to obtain $\Delta\mathbf{H}_k$ that is guaranteed to produce a positive-definite-Hessian is the key issue here. We describe in the subsection below a method for the determination of $\Delta\mathbf{H}_k$. Note that, once the perturbed Hessian, which is most frequently referred to as the *stabilized Hessian*, is available, $\Delta\mathbf{u}^k$ is found from

$$\tilde{\mathbf{H}}_k \Delta\mathbf{u}^k = -\mathbf{L}_k^T[(\nabla\nabla f)_k \Delta\mathbf{x}_0^k + (\nabla f)_k] \tag{6.39}$$

The process of finding a positive-definite $\tilde{\mathbf{H}}_k$ is termed *Hessian stabilization.* The rationale behind Hessian stabilization lies in the property that, if the eigenvalues of a $n \times n$ matrix $\mathbf{M}$ are $\{\,\mu_k\,\}_1^n$, then the eigenvalues of matrix $\mathbf{M} + \alpha\mathbf{1}$, where α is a real number and $\mathbf{1}$ is the $n \times n$ identity matrix, are $\{\,\mu_k + \alpha\,\}_1^n$. Thus, the effect of adding the isotropic matrix $\alpha\mathbf{1}$ to $\mathbf{M}$ is to shift the eigenvalues of the latter to the right of the complex plane by an amount α if $\alpha > 0$; if $\alpha < 0$, then the same isotropic matrix shifts the eigenvalues of $\mathbf{M}$ to the left of the complex plane by an amount $|\alpha|$. If the Hessian of interest is not positive-definite, this means that it has some negative eigenvalues, in which case Hessian stabilization consists in finding the right value of α in the foregoing scheme, that will shift the Hessian eigenvalues to the right of the real axis—since the Hessian is necessarily symmetric, its eigenvalues are all real—so that none of the shifted eigenvalues will lie on the left half of the real axis. Notice that, if $\alpha > 0$ is underestimated, then the associated isotropic matrix will fail to shift some of the negative Hessian eigenvalues to the right; if overestimated, then all shifted eigenvalues will lie on the right half of the real axis, but the Hessian will be overly perturbed, and the converegence will slow down.

Obviously, if we know the eigenvalues of the Hessian $\mathbf{H}_k$, then we can find the right μ_k that will shift all its eigenvalues to the right. However, computing eigenvalues is an iterative process, except for very special cases of simple matrices, and hence, we cannot rely on knowledge of those eigenvalues. We discuss below how to estimate the right amount of shift α without having to compute the Hessian eigenvalues. The basis of the procedure is a result on positive-definite matrices that we recall below.

Diagonal-Dominance in Positive-Definite Matrices

The *Gerschgorin Theorem* (Varga, 2000) establishes a region in the complex plane containing all the eigenvalues of a $n \times n$ matrix $\mathbf{A}$, defined over the complex field $\mathbb{C}$, namely,

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

According to the Gerschgorin Theorem, all the eigenvalues of $\mathbf{A}$ lie within a complex region $\mathcal{S}$, defined as the union of disks $\mathcal{D}_i$ centered at a_{ii}, with radius r_i, in the complex plane, for $i = 1, ..., n$, r_i being given by

$$r_i = \sum_{j=1, j\neq i}^{n} \mid a_{ij} \mid$$

in which $\mid \cdot \mid$ denotes the module of $(\,\cdot\,)$. The Gerschgorin Theorem is illustrated in Fig. 6.3, the region $\mathcal{S}$ thus being

$$\mathcal{S} = \bigcup_{i=1}^{n} \mathcal{D}_i$$

If $\mathbf{A}$ is symmetric and real, which is so for Hessian matrices, then its eigenvalues lie in the union of the real intervals

$$I_i = [a_{ii} - r_i, a_{ii} + r_i], \quad i = 1, 2, ..., n$$

A lower bound l of the set $\{\lambda_i\}_1^n$ of eigenvalues of $\mathbf{A}$ is, thus,

$$l \equiv \min_i \{a_{ii} - r_i\}_1^n \tag{6.40a}$$

the corresponding upper bound being

$$u \equiv \max_i \{a_{ii} + r_i\}_1^n \tag{6.40b}$$

If $\mathbf{A}$ is positive-definite, all eigenvalues of $\mathbf{A}$ must be positive, which means that the lower bound l should be positive as well. If, on the other hand, $\mathbf{A}$ is either sign-indefinite or positive-definite, but close to singular, then l can be negative.

Now we have, with the foregoing notation,

Definition 6.4.1 (Diagonal-dominance) A $n\times n$ matrix $\mathbf{A}$ is said to be *diagonally-dominant* if

$$a_{ii} \; > \; r_i$$

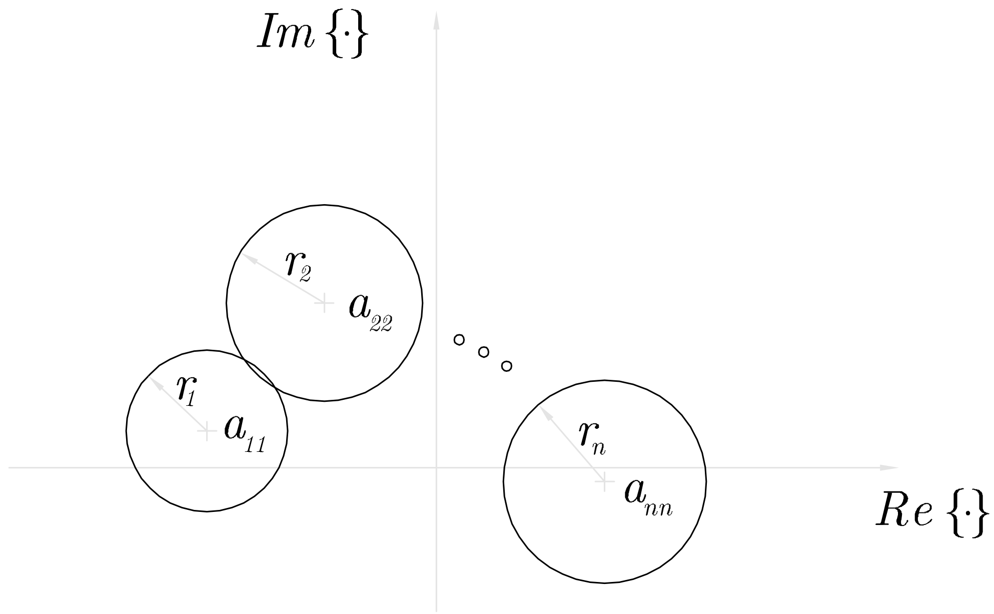


Figure 6.3: The Gerschgorin disks of an arbitrary $n \times n$ matrix

Further, we have a result allowing us to characterize positive-definite matrices without the burden of computing their eigenvalues.

Theorem 6.4.1 *If a symmetric matrix* $\mathbf{A}$ *is diagonally-dominant, then it is positive-definite.*

Note, however, that the converse is not true, i.e., a positive-definite matrix need not be diagonally-dominant.

6.4.2 Hessian Stabilization with the Aid of Diagonal-Dominance

The feasible Hessian matrix $\mathbf{L}_k^T(\nabla\nabla f)|_k\mathbf{L}_k$ of the objective function can fail to be positive-definite when the Hessian $(\nabla\nabla f)|_k$ fails to be so. However, it may well happen that the latter fails to be positive-definite and yet the former is positive-definite. In this light, it appears that we need not stabilize the Hessian itself, but only its feasible projection. In practice, we have found that stabilizing the Hessian, rather than only its feasible projection, leads to a more robust procedure. We will thus proceed accordingly.

We need first a criterion to tell us when $(\nabla\nabla f)_k$ is *suspected* of being sign-indefinite. The criterion is simple:

If $(\nabla\nabla f)_k$ fails to be diagonally-dominant, then sign-indefiniteness is likely to occur—but not guaranteed!—and hence, Hessian stabilization is warranted.

The stabilizing procedure is applied by introducing a scalar $\mu_k > 0$, that we will term the *Gerschgorin shift*, such that a new diagonally-dominant matrix $\mathbf{W}_k$ is used to replace $(\nabla\nabla f)_k$, with $\mathbf{W}_k$ defined as

$$\mathbf{W}_k = (\nabla\nabla f)_k + \mu_k \mathbf{1} \tag{6.41}$$

$\mathbf{W}_k$ thus being guaranteed to be positive-definite, based on the diagonal-dominance theorem above.

The stabilized Hessian thus yields the feasible Hessian

$$\tilde{\mathbf{H}}_k = \mathbf{L}_k^T \mathbf{W}_k \mathbf{L}_k = \mathbf{L}_k^T (\nabla\nabla f)|_k \mathbf{L}_k + \mu_k \mathbf{L}_k^T \mathbf{L}_k \tag{6.42}$$

Moreover, since $\mathbf{L}_k$ is isotropic, for it has been chosen as an isotropic orthogonal complement of $\mathbf{J}_k$, it turns out that

$$\mathbf{L}_k^T \mathbf{L}_k = \mathbf{1}_{n'} \tag{6.43}$$

with $\mathbf{1}_{n'}$ denoting the $(n-l) \times (n-l)$ identity matrix. Therefore, the *stabilized feasible Hessian* $\tilde{\mathbf{H}}_k$ reduces to

$$\tilde{\mathbf{H}}_k = \mathbf{L}_k^T \mathbf{W}_k \mathbf{L}_k = \mathbf{L}_k^T (\nabla\nabla f)|_k \mathbf{L}_k + \mu_k \mathbf{1}_{n'} \tag{6.44}$$

which is now *a fortiori* positive-definite, problem (6.36) thus admitting one minimum $\Delta\mathbf{u}^k$, which is computed from eq.(6.39).

Choice of the Gerschgorin Shift

In this subsection we stress the importance of the selection of μ_k, where subscript k denotes the iteration number. With a proper selection, the number of iterations can be effectively reduced. First, we assume that $(\nabla\nabla f)_k$ was found to fail the diagonal-dominance test, and hence, the lower bound l_k of its eigenvalues is negative.

The selection of μ_k is suggested to be *slightly* greater than the lower bound l_k of the eigenvalues, as obtained by the diagonal-dominance criterion (or test), i.e.,

$$\mu_k = -(1 + \Delta_k) l_k \tag{6.45}$$

where Δ_k is a positive number, that is to be chosen as small as possible.

The value of Δ_k is related to the bandwidth b_k of the Hessian eigenvalues, with b_k defined as

$$b_k = u_k - l_k$$

Example 6.4.1 (Powell's Function)
A problem proposed by (Powell, 1969) is solved here:

$$f(\mathbf{x}) = e^{x_1 x_2 x_3 x_4 x_5} \quad \rightarrow \quad \min_{\mathbf{x}}, \quad \mathbf{x} \equiv [\, x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \,]^T$$

subject to the nonlinear equality constraints

$$\begin{aligned} h_1 &= x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0 \\ h_2 &= x_2 x_3 + x_4 x_5 = 0 \\ h_3 &= x_1^3 + x_2^3 + 1 = 0 \end{aligned}$$

A word of caution is in order here: while the exponential function e^x is convex—its second derivative with respect to x is positive everywhere—the bivariate exponential function $e^{x_1 x_2}$ is not convex everywhere, and neither is so the above objective function. In fact, the Hessian of the bivariate exponential becomes sign-indefinite in a region of the x_1-x_2 plane. This statement is illustrated with the plot of this function displayed in Fig. 6.4. The conclusion of the foregoing discussion is, then, that the multivariable exponential function, like Powell's function, has a Hessian that is sign-indefinite in a region of $\mathbb{R}^5$. Optimum solutions were obtained with two different algorithms, the corresponding results being listed in Table 6.2. Results were obtained under the same environment, a Silicon Graphics 64-bit Octane SE workstation, with a 250 MHz R10000 processor, running the IRIX 6.5 operating system. An initial guess is taken as

$$\mathbf{x}_0 = [\, -1 \quad 2 \quad -0.5 \quad 1 \quad 2 \,]^T$$

with tolerance of 10^{-6}. The ODA package requires only 58 iterations, as compared with 186 required by the Matlab Optimization Toolbox. Moreover, the CPU time required by the ODA is only 8.9 % of the CPU time consumed by Matlab.

Example 6.4.2 (The Equilibrium Configuration of a N-link Chain)
Shown in Fig. 6.5a is a chain with N links in its equilibrium configuration, which spans a distance d, with each link of length ℓ. Knowing that the chain reaches its equilibrium configuration when its potential energy attains a minimum value, find

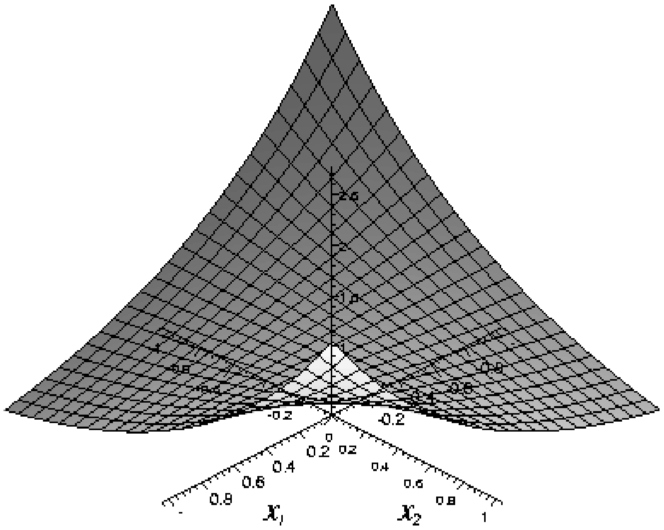


Figure 6.4: The bivariate exponential $e^{x_1 x_2}$

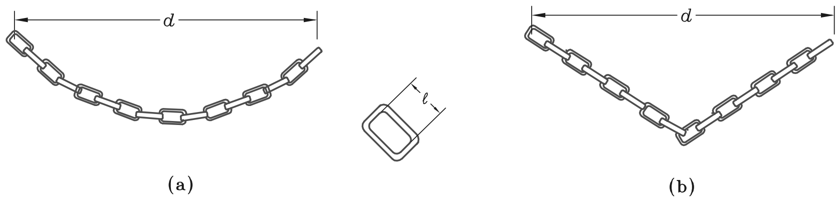


Figure 6.5: An N-link chain in: (a) its unknown equilibrium configuration; and (b) a configuration to be used as an initial guess

the said equilibrium configuration. This problem, originally proposed by Luenberger (1984), was solved for the case of two design variable, exactly, in Example 5.3.2.

Angles θ_i, used to define the configuration of the chain, are measured from the vertical ccw, with θ_i corresponding to the angle that the axis of the ith link makes with the vertical, as shown in Fig. 6.6.

If $V \equiv \mu\ell f(\theta_1, \theta_2 \ldots, \theta_N)$ denotes the potential energy of the chain, and μ is the mass density of the links per unit length, then minimizing V is equivalent to minimizing f, which is given by

$$f(\theta_1, \theta_2 \ldots, \theta_N) = -\left[\frac{1}{2}\cos\theta_1 + (\cos\theta_1 + \frac{1}{2}\cos\theta_2) + \ldots \right.$$
$$\left. + (\cos\theta_1 + \ldots + \cos\theta_{N-1} + \frac{1}{2}\cos\theta_{N-1})\right] \quad \rightarrow \quad \min_{\{\theta_i\}_1^N}$$

Table 6.2: A performance comparison based on Powell's function

	Matlab	ODA
f	0.05395	0.05395
x_1	-1.7172	-1.7171
x_2	1.5957	1.5957
x_3	1.8272	-1.8272
x_4	0.7636	-0.7636
x_5	0.7636	0.7636
# of iterations	186	55
CPU time (s)	0.2903	0.0259

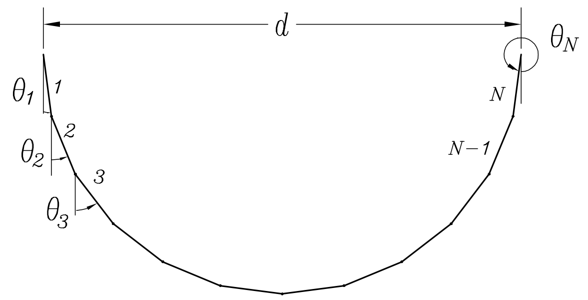


Figure 6.6: Definition of θ_i for the N-link chain

or, in compact form,

$$f(\theta_1,\, \theta_2\, \ldots,\, \theta_N) = -\frac{1}{2}\sum_{i=1}^{N}[2(N-i)+1]\cos\theta_i \quad \to \quad \min_{\{\,\theta_i\,\}_1^N}$$

subject to two constraints: the two ends (1) must lie at the same height, and (2) are separated by a distance d, as shown in Fig. 6.6. The constraints are

$$h_1 = \sum_{i=1}^{N}\cos\theta_i = 0$$

$$h_2 = \sum_{i=1}^{N}\sin\theta_i - \frac{d}{\ell} = 0$$

Under the assumption that the configuration is symmetric, and that N is even, then $M = N/2$ is an integer. Thus, only one half of the chain need be considered. The

Table 6.3: Luenberger's chain with $M = 5$

	Matlab	The ODA
θ_1	0.0893	0.0893
θ_2	0.1147	0.1147
θ_3	0.1599	0.1599
θ_4	0.2625	0.2625
θ_5	0.67856	0.6785
f_{min}	-12.2650	-12.2650
Iterations	16	7
CPU time (s)	0.2825	0.003261

problem is, thus, simplified as

$$f(\theta_1,\, \theta_2\, \ldots,\, \theta_M) = -\left[\frac{1}{2}\cos\theta_1 + (\cos\theta_1 + \frac{1}{2}\cos\theta_2) + \ldots + (\cos\theta_1 + \ldots + \cos\theta_{M-1}\right.$$
$$\left. + \frac{1}{2}\cos\theta_M)\right] = -\frac{1}{2}\sum_{i=1}^{M}[2(M-i)+1]\cos\theta_i \quad \rightarrow \quad \min_{\{\cos\theta_i\}_1^M}$$

The two constraints then reduce to only one:

$$h = \sum_{i=1}^{M}\sin\theta_i - \frac{d}{2\ell} = 0$$

This problem, with $M = 5$, i.e., with $N = 10$, is solved now using the configuration of Fig. 6.5b as an initial guess. The equilibrium configuration of the chain is given in Table 6.3 with a comparison between ODA and Matlab.

With the same tolerance set at 0.0001, the ODA takes less than half the number of iterations than Matlab; additionally, the CPU time consumed by ODA is about 10% of that consumed by Matlab.

It is noteworthy that the convergence of ODA is dependent on the choice of μ_k in eq.(6.41), for a given value of $d/(2l)$.

Example 6.4.3 (Minimum value of the Rosenbrock Function) *We include an example where the ODA package is used to find the minimum value of the Rosenbrock function (Rosenbrock, 1960), a.k.a. the* banana function, *defined as*

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{6.46}$$

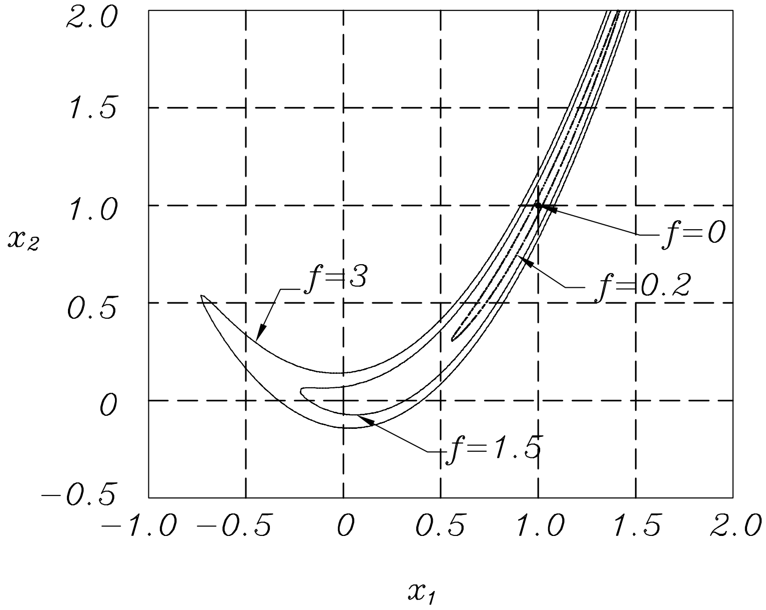


Figure 6.7: The contours of the Rosenbrock (a.k.a. the banana) function

The problem can be treated as finding an "approximate" solution of a system of nonlinear equations, namely,

$$\boldsymbol{\phi} = \begin{bmatrix} x_2 - x_1^2 \\ 1 - x_1 \end{bmatrix} \tag{6.47}$$

such that the least-square error f is a minimum, i.e.,

$$f(\mathbf{x}) = \frac{1}{2}\boldsymbol{\phi}^T\mathbf{W}\boldsymbol{\phi} \quad \rightarrow \quad \min_{x_1,\,x_2} \tag{6.48}$$

with

$$\mathbf{W} = \begin{bmatrix} 200.0 & 0 \\ 0 & 2.0 \end{bmatrix} \tag{6.49}$$

By taking $\mathbf{x}_0 = [0.2 \quad 0.2]^T$ as initial guess, we obtained the sequence of values $\mathbf{x}^1, \mathbf{x}^2, \ldots$ shown in Table 6.4. The optimum was reached after 3 iterations, and found to be $\mathbf{x}_{opt} = [1 \quad 1]^T$. The contours of the banana function are plotted in Fig. 6.7. Now, since $\boldsymbol{\phi} = \mathbf{0}$ at $\mathbf{x}_{opt}$, the normality condition (3.95) is readily verified.

Table 6.4: Interations toward the minimum of the Rosenbrock function

i	1	2	3
$\mathbf{x}^i$	$[1.000000,\ 0.360000]^T$	$[1.000000,\ 1.000000]^T$	$[1.000000,\ 1.000000]^T$

Example 6.4.4 (The Constrained Minimization of the Rosenbrock Function) *In this example, we find its equality-constrained minimum of the Rosenbrock function using SQP via the ODA. We thus have*

$$f(x_1,\, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \quad \rightarrow \quad \min_{x_1,\, x_2}$$

subject to

$$\begin{aligned} h((x_1,\, x_2) = {} & 0.7525x_1^2 - 1.1202x_1 - 0.8574x_1x_2 \\ & +0.6168x_2 + 0.2575x_2^2 + 0.4053 \end{aligned}$$

The function is notorious for its ill-conditioning, which is apparent from its contours, as shown in Fig. 6.8, showing elongated valleys. The outcome is that the quadratic approximation of this function within those valleys is a family of ellipses that have one semiaxis much greater than the other one, thereby leading to ill-conditioning. Notice that the constraint is a rather elongated ellipse that contributes to the ill-conditioning of the problem.

Starting from the initial guess $\mathbf{x} = [1.5\ 1.5]^T$ *with a damping ratio of 0.025, the optimum solution is found in 312 ODA iterations, the result being*

$$\mathbf{x}_{opt} = \begin{bmatrix} 0.9176 \\ 0.5873 \end{bmatrix}$$

which yields

$$f_{\min} = 6.6963$$

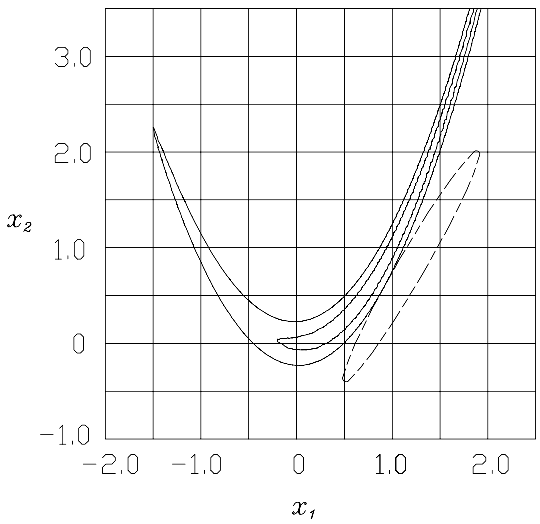


Figure 6.8: The contours of the Rosenbrock (banana) function and its quadratic constraint (dashed)

Chapter 7

Inequality-Constrained Optimization

7.1 Introduction

The constraints under which a design problem is formulated involve, more often than not, inequality constraints, in addition to equalities. In fact, inequality constraints arise naturally in design because the resources available to accomplish a design job are *finite*. For example, a designed object must: fit into a designated region; be realizable within a given budget; and be delivered by a certain date.

In this chapter we address two issues around inequality-constrained problems: the *normality conditions* and the *methods of solution*. As to the former, we will not dwell into their rigorous derivation, which are elusive to a simple analysis with the tools of linear algebra; rather, we will introduce the first-order normality conditions without derivation, and illustrate their validity with examples. The second-order normality conditions will be derived using an intuitive approach, more so than a mathematical formulation.

The reason why linear algebra is no longer sufficient to derive the normality conditions of inequality-constrained problems lies in the nature of inequalities, which define a *region* of $\mathbb{R}^n$ that is neither a vector subspace nor a manifold, as we encountered when studying equality-constrained problems. Now we will speak, more generally, of the *feasible region* $\mathcal{R}_F$, which can have sharp edges and vertices, not occurring in manifolds. For this reason, a simple transformation of the form $\mathbf{x}(\mathbf{u})$ is not sufficient, in general, to guarantee the fulfillment of the inequality constraints.

7.2 The Karush-Kuhn-Tucker Conditions

The first-order normality conditions of equality-constrained problems are classical results, first proposed by Joseph Louis de Lagrange, brilliant mathematician born in Turin in 1736 and dead in Paris in 1813. Lagrange founded in Turin a society that would become the Academy of Sciences; then, Lagrange went to Berlin, to the Academy of Friedrich II, to succeed Euler. Rather late in his life, in 1787, did Lagrange move to Paris, invited by Louis XVI to teach at *Ecole normale*. Appointed senator and made count by Napoleon, Lagrange became one of the first professors at *Ecole polytechnique*.

The first-order normality conditions for inequality-constrained problems had to wait until well into the XX century. These conditions were disclosed first by W. Karush in his M.S. thesis in the Department of Mathematics at the University of Chicago (Karush, 1939). Apparently, these results were never published in the archival literature, for which reason they remained unknown. Twelve years later, they were published in the *Proc. Second Berkeley Symposium* by Kuhn and Tucker (1951). The credit of these normality conditions has gone mostly to Kuhn and Tucker, but given their history, these conditions are sometimes referred to as the *Karush-Kuhn-Tucker conditions*.

The problem at hand is formulated as

$$f(\mathbf{x}) \quad \to \quad \min_{\mathbf{x}} \tag{7.1a}$$

subject to

$$\mathbf{g}(\mathbf{x}) \le \mathbf{0}_p \tag{7.1b}$$

$$\mathbf{h}(\mathbf{x}) = \mathbf{0}_l \tag{7.1c}$$

where inequality (7.1b) is to be taken with a grain of salt: Arrays *not forming ordered sets*, this relation has no verbatim meaning. It is to be interpreted as shorthand for a set of m inequalities, namely,

$$\begin{aligned} g_1(\mathbf{x}) &\equiv g_1(x_1,\, x_2,\, \dots,\, x_n) \le 0 \\ g_2(\mathbf{x}) &\equiv g_2(x_1,\, x_2,\, \dots,\, x_n) \le 0 \\ &\vdots \\ g_p(\mathbf{x}) &\equiv g_p(x_1,\, x_2,\, \dots,\, x_n) \le 0 \end{aligned}$$

To formulate the normality conditions, we proceed as before, namely, by defining a

Lagrangian upon adjoining the equality and the inequality constraints to the objective function, namely,

$$F(\mathbf{x},\, \boldsymbol{\lambda},\, \boldsymbol{\mu}) \equiv f(\mathbf{x}) + \boldsymbol{\lambda}^T\mathbf{h}(\mathbf{x}) + \boldsymbol{\mu}^T\mathbf{g}(\mathbf{x}) \quad \rightarrow \quad \min_{\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\mu}} \tag{7.2}$$

where $\boldsymbol{\lambda}$ is found so that $\mathbf{h}(\mathbf{x})$ will vanish at a stationary point, the role played by $\boldsymbol{\mu}$ being described below. We start by introducing a

Definition 7.2.1 (Active constraint) *When, at a stationary point, the ith constraint of (7.1b) holds with the equality sign, this constraint is said to be* active.

By extension, we will define as *passive* any non-active constraint.

The components of $\boldsymbol{\mu}$ are chosen so that the contribution of all passive constraints to the Lagrangian vanish, that of all active constraints necessarily vanishing.

While the normality conditions cannot be derived by simply making the gradient of the foregoing Lagrangian equal to zero, these conditions look very much like those associated with equality-constrained problems. Indeed, $\mathbf{x}_0$ is a *feasible* stationary point if

$$\mathbf{h}(\mathbf{x}_0) = \mathbf{0}_l, \quad \mathbf{g}(\mathbf{x}_0) \leq \mathbf{0}_p \tag{7.3a}$$

$$\nabla f|_{\mathbf{x}=\mathbf{x}_0} + \mathbf{J}_0^T\boldsymbol{\lambda} + \mathbf{G}_0^T\boldsymbol{\mu} = \mathbf{0}_n \tag{7.3b}$$

$$\boldsymbol{\mu} \geq \mathbf{0}_m, \quad \boldsymbol{\mu}^T\mathbf{g}(\mathbf{x}) = 0 \tag{7.3c}$$

where

$$\mathbf{G}_0 \equiv \mathbf{G}(\mathbf{x}_0) \equiv \left.\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0}, \quad \mathbf{J}_0 \equiv \mathbf{J}(\mathbf{x}_0) \equiv \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_0} \tag{7.3d}$$

i.e., $\mathbf{G}$ and $\mathbf{J}$ are, respectively, the $p \times n$ and the $l \times n$ gradients of the inequality- and equality-constraint functions $\mathbf{g}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$, namely,

$$\mathbf{G} \equiv \begin{bmatrix} (\nabla g_1)^T \\ (\nabla g_2)^T \\ \vdots \\ (\nabla g_p)^T \end{bmatrix}, \quad \mathbf{J} \equiv \begin{bmatrix} (\nabla h_1)^T \\ (\nabla h_2)^T \\ \vdots \\ (\nabla h_l)^T \end{bmatrix} \tag{7.4}$$

Hence, the KKT condition (7.3b) can be expressed alternatively as

$$\begin{aligned} \nabla f|_{\mathbf{x}=\mathbf{x}_0} &+ \lambda_1 \nabla h_1|_{\mathbf{x}=\mathbf{x}_0} + \lambda_2 \nabla h_2|_{\mathbf{x}=\mathbf{x}_0} + \cdots + \lambda_1 \nabla h_l|_{\mathbf{x}=\mathbf{x}_0} \\ &+ \mu_1 \nabla g_1|_{\mathbf{x}=\mathbf{x}_0} + \mu_2 \nabla g_2|_{\mathbf{x}=\mathbf{x}_0} + \cdots + \mu_p \nabla g_p|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{0}_n \end{aligned} \tag{7.5}$$

It is noteworthy that, the components of $\boldsymbol{\mu}$ being non-negative and those of $\mathbf{g}(\mathbf{x})$ non-positive, each $\mu_i g_i(\mathbf{x})$ is non-positive. Hence, $\boldsymbol{\mu}^T\mathbf{g}(\mathbf{x}) = 0$ of eq.(7.3c) implies

$$\mu_i g_i(\mathbf{x}) = 0, \quad i = 1, \ldots, p$$

at a feasible stationary point.

Relations (7.3a–c) are the *Karush-Kuhn-Tucker* (KKT) *conditions.* These are the first-order normality conditions of the inequality-constrained problem at hand, and hence, guarantee a feasible stationary point of $\mathcal{R}_F$, but not a minimum. The latter is guaranteed by the second-order normality conditions, to be studied in Section 7.3.
Remark: In the absence of inequality constraints, eq.(7.3b) reduces to the FONC of equality-constrained problems, eq.(5.8a).

In the third row of the KKT, eqs.(7.3c), the third of these relations is referred to as the *complementary slackness*, sometimes as the *transversality condition*[1]. Complementary slackness thus guarantees that, at a SP, the Lagrangian equals the objective function, under the assumption that the equality constraints are verified.

Further, if a of the m inequality constraints are active, we can partition vector $\mathbf{g}(\mathbf{x}_0)$, very likely after a reshuffling of its components, in the form

$$\mathbf{g}(\mathbf{x}_0) = \begin{bmatrix} \mathbf{g}_a \\ \mathbf{g}_{p'} \end{bmatrix}, \quad p' = p - a \tag{7.6}$$

where $\mathbf{g}_a$ and $\mathbf{g}_{p'}$ are a- and $(p-a)$-dimensional vectors, respectively. Now, the Karush-Kuhn-Tucker conditions of eq.(7.5) can be restated as

$$\begin{aligned} \nabla f|_{\mathbf{x}=\mathbf{x}_0} &+ \lambda_1 \nabla h_1|_{\mathbf{x}=\mathbf{x}_0} + \lambda_2 \nabla h_2|_{\mathbf{x}=\mathbf{x}_0} + \cdots + \lambda_l \nabla h_l|_{\mathbf{x}=\mathbf{x}_0} \\ &+ \mu_1 \nabla g_1|_{\mathbf{x}=\mathbf{x}_0} + \mu_2 \nabla g_2|_{\mathbf{x}=\mathbf{x}_0} + \cdots + \mu_a \nabla g_a|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{0}_n,\ \mathbf{g}_a = \mathbf{0}_a \end{aligned} \tag{7.7}$$

where the first equation resembles the FONC of equality-constrained problems, this time with $l + a$ equality constraints, as derived in eq.(5.8a).

Moreover, the KKT conditions can be cast in the *canonical form*

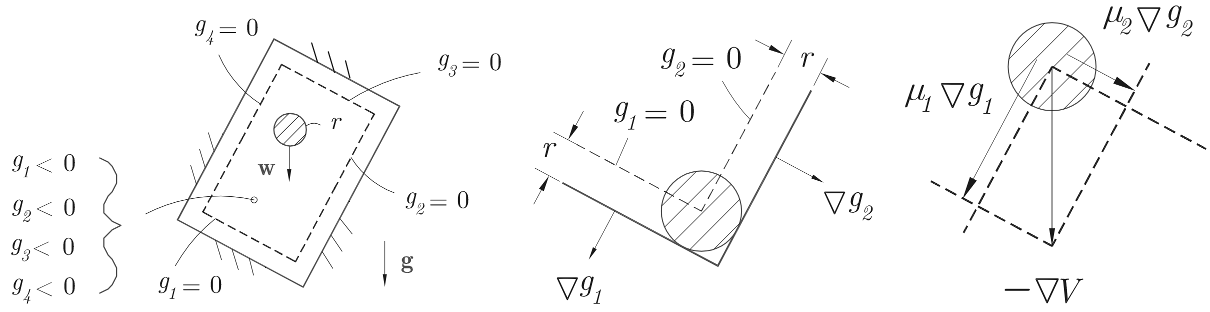
$$\nabla_{\mathbf{x}} F \equiv \frac{\partial F}{\partial \mathbf{x}} = \mathbf{0}_n, \quad \nabla_{\boldsymbol{\mu}} F \leq \mathbf{0}_p, \quad \boldsymbol{\mu}^T \nabla_{\boldsymbol{\mu}} F = 0 \tag{7.8}$$

A proof of the KKT can be found in (Culioli, 1994; Boyd and Vandenberghe, 2004).

To illustrate the validity of the KKT conditions, we give in Fig. 7.1 a mechanical interpretation: A heavy ball of weight $\mathbf{w}$ is constrained to lie in a box, under the

[1] Not to be confused with the transversality condition of *calculus of variations.*

action of the gravity field $\mathbf{g}$, as depicted in Fig. 7.1a; the ball is shown in its equilibrium position in Fig. 7.1b; in Fig. 7.1c, the weight of the ball, equal to $-\nabla V$, where V is the potential energy of the ball, is decomposed into the two forces normal to the box walls. Notice that these two components push the walls, but cannot pull them, which is the reason why $\mu_i > 0$, for $i = 1, 2$.



(a) The general layout

(b) The ball at its equilibrium position

(c) A decomposition of the weight of the ball at equilibrium position

Figure 7.1: A heavy ball inside a box inclined with respect to the vertical

Example 7.2.1

Consider the problem

$$f = \frac{1}{2}(x_1^2 + x_2^2) \quad \rightarrow \quad \min_{x_1, x_2}$$

subject to

$$x_1 + x_2 \geq 10$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$

The objective function and the constraints are illustrated in Fig. 7.2.

For starters, we must express the inequality constraints in the standard form adopted at the outset, i.e., as $g_i(\mathbf{x}) \leq 0$, whence,

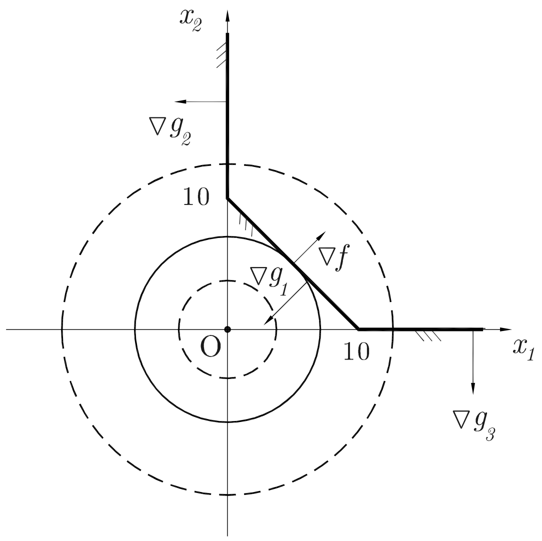


Figure 7.2: A quadratic objective function subject to linear inequality constraints

$$g_1 \equiv -x_1 - x_2 + 10 \leq 0, \quad g_2 \equiv -x_1 \leq 0, \quad g_3 \equiv -x_2 \leq 0$$

Apparently, the minimum is found at $\mathbf{x}_0 = [5,\, 5]^T$. *We evaluate then the items entering in the KKT conditions at this point* $\mathbf{x}_0$*:*

$$\nabla g_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \nabla g_2 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad \nabla g_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$
$$\nabla f = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \Rightarrow \quad \nabla f|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$$

By inspection, only the first constraint is active, and hence,

$$g_1(\mathbf{x}_0) = 0, \quad \mu_1 > 0, \quad \mu_2 = \mu_3 = 0$$

The KKT condition (7.7) thus reduces to

$$\nabla f|_{\mathbf{x}=\mathbf{x}_0} + \mu_1 \nabla g_1|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{0}_2, \quad \mu_1 > 0$$

or

$$\mu_1 \nabla g_1|_{\mathbf{x}=\mathbf{x}_0} = -\left.\nabla f\right|_{\mathbf{x}=\mathbf{x}_0}$$

which states that, at the SP $\mathbf{x}_0$ *given above, the two gradients,* $\nabla f|_{\mathbf{x}=\mathbf{x}_0}$ *and* $\nabla g_1|_{\mathbf{x}=\mathbf{x}_0}$*, are linearly-dependent. As a consequence, the above overdetermined system of two equations in one single unknown,* μ_1*, admits one solution that verifies the two equations. Upon solving this system, in fact, we obtain* $\mu_1 = 5 > 0$*, thereby verifying the second relation of conditions (7.3c).*

Example 7.2.2

$$f \equiv 8x_1^2 - 8x_1x_2 + 3x_2^2 \quad \rightarrow \quad \min_{\mathbf{x}}$$

subject to

$$x_1 \geq 3$$
$$x_2 \leq \frac{3}{2}$$

The objective function and the constraints of this example are depicted in Fig. 7.3

Again, we start by restating the inequalities in our standard form:

$$g_1 \equiv 3 - x_1 \leq 0, \; g_2 \equiv x_2 - \frac{3}{2}$$

Therefore,

$$\nabla g_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \; \nabla g_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

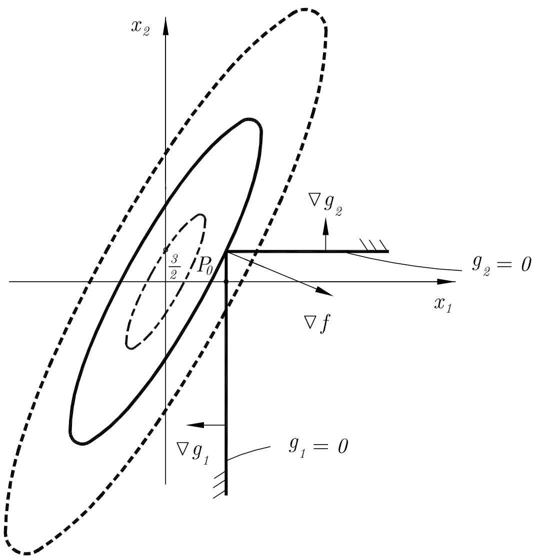


Figure 7.3: One more quadratic objective function subject to linear inequality constraints

condition (7.3b) thus leading to

$$\mu_1 \nabla g_1|_{\mathbf{x}=\mathbf{x}_0} + \mu_2 \nabla g_2|_{\mathbf{x}=\mathbf{x}_0} = -\nabla f|_{\mathbf{x}=\mathbf{x}_0}$$

where, apparently,

$$\mathbf{x}_0 = \begin{bmatrix} 3 \\ 3/2 \end{bmatrix}, \quad \nabla f = \begin{bmatrix} 16x_1 - 8x_2 \\ -8x_1 + 6x_2 \end{bmatrix}$$

Hence,

$$\nabla f|_{\mathbf{x}=\mathbf{x}_0} = \begin{bmatrix} 36 \\ -15 \end{bmatrix}$$

The above normality condition thus leading to

$$\mu_1 \begin{bmatrix} -1 \\ 0 \end{bmatrix} + \mu_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -36 \\ 15 \end{bmatrix}$$

which, in this case, turns out to be a determined system of two equations in two unknowns, its solution being

$$\mu_1 = 36 > 0, \quad \mu_2 = 15 > 0$$

thereby verifying all KKT conditions.

Example 7.2.3 (A Linear Program)

A class of optimization problems finding a number of applications involves a linear objective function subject to linear equality and inequality constraints. This class is studied within the realm of linear programming. *These problems cannot be solved with the tools described so far, for we have focused on least-square problems, with an extension to more general objective functions and equality constraints. By the same token, linear programs arise seldom in mechanical design. To be true, a family of design problems in structural engineering, known as* limit design, *pertain to the design of structural elements, beams, columns and plates, for minimum weight, in such a way that all modes of* plastic failure *are avoided. Problems in limit design lead to linear programs.*

Linear programming is a first instance of application of the KKT conditions. We illustrate the concept with the problem below.

$$f \equiv 2x_1 - x_2 \quad \rightarrow \quad \min_{x_1, x_2}$$

subject to

$$\begin{aligned} g_1(\mathbf{x}) &\equiv -x_1 \leq 0 \\ g_2(\mathbf{x}) &\equiv -x_2 \leq 0 \\ g_3(\mathbf{x}) &\equiv x_1 + x_2 - 1 \leq 0 \end{aligned}$$

The objective function and the constraints of this problem are illustrated in Fig. 7.4.

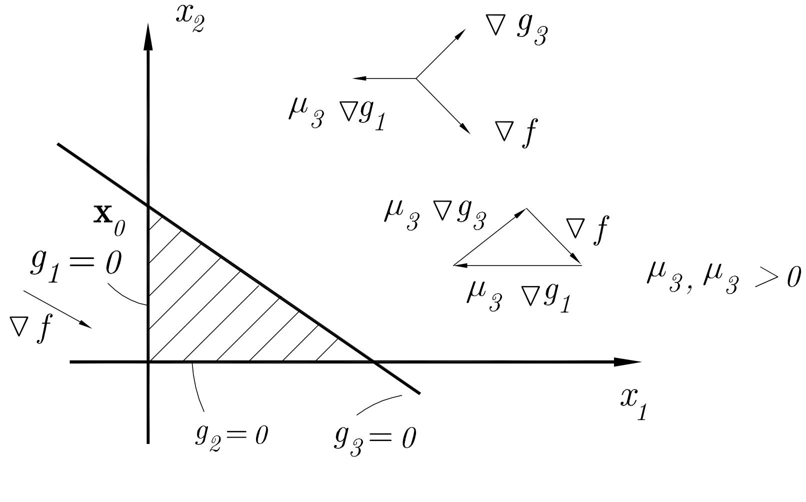


Figure 7.4: A linear program

In this case,

$$\mathbf{G} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 1 \end{bmatrix}, \quad \nabla f = \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \quad \mathbf{x}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The KKT conditions lead to

$$\begin{bmatrix} 2 \\ -1 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Apparently, g_1 and g_3 are active, and hence,

$$\mu_1,\ \mu_3 > 0, \quad \mu_2 = 0$$

thereby ending up with a system of two equations in two unknowns, μ_1 and μ_3. Upon solving this system, we obtain, successively,

$$\mu_3 = 1 > 0, \quad \mu_1 = 3 > 0$$

thereby verifying the KKT conditions.

7.3 Second-Order Normality Conditions

The simplest way of stating the *sufficient* conditions for a minimum, i.e., the second-order normality conditions, is by imposing the condition that, at a stationary point $\mathbf{x}_0$ within $\mathcal{R}_F$, any *feasible move* $\Delta\mathbf{x}_F$ will produce a *growth*, and hence, a *worsening*, of the objective function while respecting the constraints. In other words, at a feasible minimum, we cannot decrease the objective function without violating the constraints. That is,

$$\Delta f \equiv f(\mathbf{x}_0 + \Delta\mathbf{x}_F) - f(\mathbf{x}_0) > 0 \tag{7.9}$$

$$\Delta\mathbf{h} \equiv \mathbf{h}(\mathbf{x}_0 + \Delta\mathbf{x}_F) - \mathbf{h}(\mathbf{x}_0) = \mathbf{0}_l \tag{7.10}$$

$$\mathbf{g}(\mathbf{x}_0 + \Delta\mathbf{x}_F) \le \mathbf{0}_p \tag{7.11}$$

A feasible move, moreover, is to be understood here at the first-order approximation of the objective function and the constraint functions $\mathbf{g}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$. In this light, then, eq.(7.10) constrains $\Delta\mathbf{x}_F$ to lie in the nullspace of $\mathbf{J}$, i.e.,

$$\mathbf{J}\Delta\mathbf{x}_F = \mathbf{0}_l \tag{7.12}$$

If we recall now the partitioning of $\mathbf{g}(\mathbf{x})$ introduced in eq.(7.6), relation (7.11) can be correspondingly partitioned as

$$\Delta\mathbf{g}_a(\mathbf{x}) \equiv \mathbf{g}_a(\mathbf{x}_0 + \Delta\mathbf{x}_F) - \mathbf{g}_a(\mathbf{x}_0) = \mathbf{0}_a \tag{7.13a}$$

$$\mathbf{g}_{p'}(\mathbf{x}) \equiv \mathbf{g}_{p'}(\mathbf{x}_0 + \Delta\mathbf{x}_F) < \mathbf{0}_{p'} \tag{7.13b}$$

the feasible move thus requiring that the passive constraints remain passive and that their active counterparts remain active. If we now let $\mathbf{G}_a$ denote $\nabla\mathbf{g}_a$, the first-order approximation of eq.(7.13a) leads to

$$\mathbf{G}_a(\mathbf{x})\Delta\mathbf{x}_F = \mathbf{0}_a \tag{7.14}$$

and we need not worry about the passive constraints (7.13b), which will be respected as long as $\|\Delta\mathbf{x}_F\|$ is "small enough." Now we can adjoin eq.(7.14) to eq.(7.12) in the form

$$\overline{\mathbf{J}}\Delta\mathbf{x}_F = \mathbf{0}_{l+a}, \quad \overline{\mathbf{J}} \equiv \begin{bmatrix} \mathbf{J} \\ \mathbf{G}_a \end{bmatrix} \tag{7.15}$$

Further, we introduce, correspondingly, a $n \times (n-l-a)$ orthogonal complement $\overline{\mathbf{L}}$ of $\overline{\mathbf{J}}$, i.e.,

$$\overline{\mathbf{J}}\,\overline{\mathbf{L}} = \mathbf{O}_{(l+a)\times(n-l-a)} \tag{7.16}$$

The *sufficient* second-order normality conditions can now be stated in exactly the same form as for equality-constrained problems. That is, we define now the $(n-l-a)\times(n-l-a)$ *feasible Hessian* as

$$\overline{\nabla\nabla} f \equiv \overline{\mathbf{L}}^T(\nabla\nabla f)\overline{\mathbf{L}}$$

and hence, the sufficient SONC can be stated as:

> *A stationary point of an inequality-constrained problem is a minimum if the feasible Hessian is positive-definite.*

7.3.1 Overconstrained Problems

In the special case in which $l + a > n$, the orthogonal complement $\overline{\mathbf{L}}$ does not exist for a full-rank $\overline{\mathbf{J}}$. In this case, the problem is overconstrained, and hence, ill-defined. However, if $\overline{\mathbf{J}}$ is rank-deficient, of rank smaller than n, then some of the constraints are redundant, at least to a first order, and the problem may admit a minimum. If $l + a = n$, then the equality constraints yield a determined system of nonlinear equations, which can be solved using the Newton-Raphson method. Any of the solutions thus obtained is a solution candidate for the original optimization problem.

Example 7.3.1 The KKT and Second-Order Conditions
Consider the problem

$$f(\mathbf{x}) = (x_1 - 1)^2 + (x_2 + 2)^2 \;\rightarrow\; \min_{x_1, x_2}$$

subject to

$$g(\mathbf{x}) = x_1 - x_2 + 0.5 \le 0$$

Solution: *We start by finding a feasible stationary point* $\mathbf{x}_0$ *via the KKT conditions:*

$$\nabla f + \mu \nabla g = 0$$

where

$$\nabla f = 2 \begin{bmatrix} x_1 - 1 \\ x_2 + 2 \end{bmatrix}, \quad \nabla g = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Hence, the KKT conditions lead to

$$2 \begin{bmatrix} x_1 - 1 \\ x_2 + 2 \end{bmatrix} + \mu \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Now, if we assume that g *is active, the above equation yields, with* $\mu > 0$,

$$\begin{aligned} 2(x_1 - 1) + \mu &= 0 \\ 2(x_2 + 2) - \mu &= 0 \end{aligned}$$

which, upon summation, lead to

$$x_1 + x_2 = -1$$

while the inequality constraint, written as an active constraint, leads in turn to

$$x_1 - x_2 = -0.5$$

The solution of the two foregoing equations is $x_1 = -0.25,\; x_2 = -0.75$, *whence* $\mu = 2.5 > 0$ *and hence, the KKT conditions are verified. Now we verify the second-order sufficient conditions: In our case,*

$$\overline{\mathbf{J}} = [(\nabla g_a)^T] = [\; 1 \quad -1 \;] \;\Rightarrow\; \overline{\mathbf{L}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

and

$$\nabla\nabla f = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

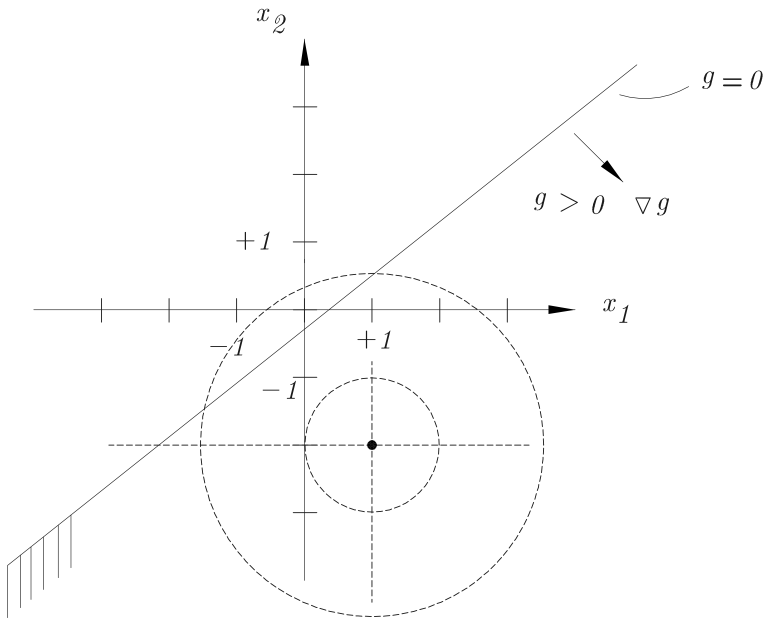


Figure 7.5: A quadratic objective function to be minimized under one equality and one inequality constraint

Therefore,

$$\overline{\nabla\nabla} f = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 4 > 0$$

thereby verifying the second-order sufficient condition. Hence, $\mathbf{x}_0 = \begin{bmatrix} -0.25 & -0.75 \end{bmatrix}^T$ *is indeed a minimum.*

Notice that, in this case, $\nabla\nabla f$ *happens to be positive-definite, and hence, the feasible Hessian is bound to be positive-definite as well. The computation of* $\overline{\nabla\nabla} f$ *in this case could thus have been dispensed with.*

7.4 Methods of Solution

Two classes of methods are available to solve inequality-constrained problems: a) *direct methods*, which handle the inequalities as such, and b) *indirect methods*, which transform the problem into one of two types, either unconstrained or equality-constrained. Indirect methods being simpler to implement, we will focus on these, which we will study first. Direct methods will be outlined at the end of the chapter.

Inequality-constrained problems can be solved using the approach introduced for either unconstrained or equality-constrained problems, upon converting the problem at hand into an unconstrained or, correspondingly, an equality-constrained problem. This can be done by various methods; we focus on two, namely, slack variables and penalty functions.

7.5 Indirect Methods

In this section, the methods of *slack variables* and of *penalty functions* are discussed.

7.5.1 Slack Variables

Upon introducing the *slack variables* $s_1, s_2, \cdots, s_p$ into inequalities (7.1b), we convert these inequalities into equality constraints, namely,

$$\boldsymbol{\gamma}(\mathbf{x}, \mathbf{s}) \equiv \begin{bmatrix} g_1 + s_1^2 \\ g_2 + s_2^2 \\ \vdots \\ g_p + s_p^2 \end{bmatrix} = \mathbf{0}, \quad \mathbf{x} \equiv \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{s} \equiv \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_p \end{bmatrix} \tag{7.17}$$

Notice that the slack variables being unknown, they have to be treated as additional design variables, the dimension of the design space being correspondingly increased. In consequence, the design vector is now of dimension $n + p$, i.e.,

$$\boldsymbol{\xi} \equiv \begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix} \tag{7.18}$$

Now, the gradient of the objective function with respect to the new design-variable vector takes the form

$$\nabla_{\boldsymbol{\xi}} f \equiv \begin{bmatrix} \nabla f \\ \nabla_{\mathbf{s}} f \end{bmatrix} \tag{7.19a}$$

where

$$\nabla f \equiv \frac{\partial f}{\partial \mathbf{x}}, \quad \nabla_{\mathbf{s}} f \equiv \frac{\partial f}{\partial \mathbf{s}} = \mathbf{0}_p \tag{7.19b}$$

the second relation following because the slack variable do not appear explicitly in the objective function.

Likewise, the Hessian with respect to the new design-variable vector $\boldsymbol{\xi}$ takes the form

$$\nabla_{\boldsymbol{\xi}} \nabla_{\boldsymbol{\xi}} f = \begin{bmatrix} \nabla\nabla f & \nabla(\nabla_{\mathbf{s}} f) \\ \nabla_{\mathbf{s}}(\nabla f) & \nabla_{\mathbf{s}}\nabla_{\mathbf{s}} f \end{bmatrix} \tag{7.20a}$$

with the notation

$$\nabla(\nabla_{\mathbf{s}} f) \equiv \frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{s}} = \left(\frac{\partial^2 f}{\partial \mathbf{s} \partial \mathbf{x}}\right)^T \equiv [\nabla_{\mathbf{s}}(\nabla f)]^T, \quad \nabla_{\mathbf{s}}\nabla_{\mathbf{s}} f \equiv \frac{\partial^2 f}{\partial \mathbf{s}^2} \tag{7.20b}$$

However, since $\nabla_{\mathbf{s}} f = \mathbf{0}_p$, the above Hessian expression reduces to

$$\nabla_{\boldsymbol{\xi}}\nabla_{\boldsymbol{\xi}} f = \begin{bmatrix} \nabla\nabla f & \mathbf{O}_{np} \\ \mathbf{O}_{np}^T & \mathbf{O}_p \end{bmatrix} \tag{7.21}$$

That is, the Hessian of the objective function with respect to the new design-variable vector $\boldsymbol{\xi}$ is singular. In case $\nabla\nabla f$ is positive-definite, $\nabla_{\boldsymbol{\xi}}\nabla_{\boldsymbol{\xi}} f$ is positive-semidefinite. Hence, in applying the method of slack variables to solve inequality-constrained problems, Hessian stabilization—see Section 6.4.1—will always be needed.

Now, the problem can be formulated as an equality-constrained problem, if we adjoin the p inequalities (7.17) to the original l, thereby obtaining a new set of equality constraints:

$$\mathbf{h}(\boldsymbol{\xi}) = \mathbf{0}_{l+p} \tag{7.22}$$

therefore, the problem at hand can be solved using ODA.

Example 7.5.1 (Minimization of the Design Error of a Four-Bar Linkage with an Input Crank)

Determine the link-lengths of the four-bar linkage shown in Fig. 7.6, that will produce the set of input-output pairs $\{\psi_i, \phi_i\}_1^q$ *shown in Table 7.1, where* ψ *and* ϕ *denote the input and output angles.*

Table 7.1: The input-output pairs of $\{\psi_i, \phi_i\}_1^{10}$

i	1	2	3	4	5
ψ_i	123.8668°	130.5335°	137.2001°	143.8668°	150.5335°
ϕ_i	91.7157°	91.9935°	92.8268°	94.2157∘	96.1601°
i	6	7	8	9	10
ψ_i	157.2001°	163.8668°	170.5335°	177.2001°	183.8668°
ϕ_i	98.6601°	101.7157°	105.3268°	109.4935°	114.2157°

The link-lengths are obtained via the Freudenstein parameters k_1, k_2 *and* k_3, *defined as*

$$k_1 = \frac{a_1^2 + a_2^2 - a_3^2 + a_4^2}{2a_2a_4}, \quad k_2 = \frac{a_1}{a_2}, \quad k_3 = \frac{a_1}{a_4} \tag{7.23a}$$

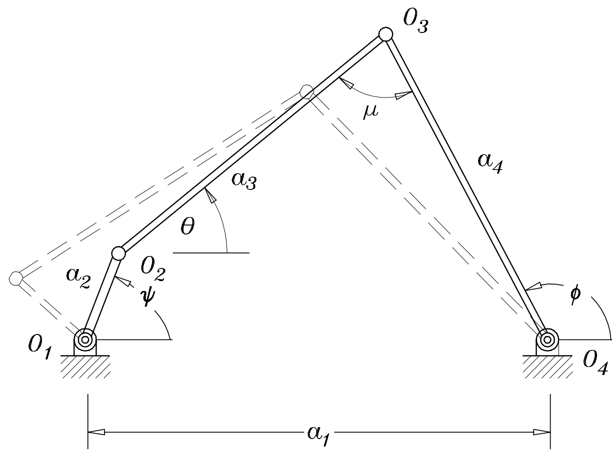


Figure 7.6: A four-bar linkage

with the inverse relations

$$a_2 = \frac{a_1}{k_1}, \quad a_3 = \frac{\sqrt{k_2^2 + k_3^2 + k_2^2 k_3^2 - 2k_1 k_2 k_3}}{|k_2 k_3|}, \quad a_4 = \frac{a_1}{k_3} \tag{7.23b}$$

for a given value of a_1. *The synthesis equations for the planar four-bar linkage can be written in the form: (Liu and Angeles, 1992)*

$$\mathbf{Sk} = \mathbf{b} \tag{7.24}$$

where $\mathbf{S}$ *is the* synthesis matrix, *and* $\mathbf{k}$ *is the vector of linkage parameters. Moreover,* $\mathbf{S}$, $\mathbf{k}$ *and* $\mathbf{b}$ *are defined as*

$$\mathbf{S} = \begin{bmatrix} 1 & \cos\phi_1 & -\cos\psi_1 \\ 1 & \cos\phi_2 & -\cos\psi_2 \\ \vdots & \vdots & \vdots \\ 1 & \cos\phi_q & -\cos\psi_q \end{bmatrix} \quad \mathbf{k} = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} \cos(\psi_1 - \phi_1) \\ \cos(\psi_2 - \phi_2) \\ \vdots \\ \cos(\psi_q - \phi_q) \end{bmatrix} \tag{7.25}$$

The design error *is defined as*

$$\mathbf{d} \equiv \mathbf{b} - \mathbf{Sk} \tag{7.26}$$

the purpose of the optimization exercise being to minimize the Euclidean norm of the design error, while ensuring that its input link is a crank. The conditions for full mobility *of the input crank are (Liu and Angeles, 1992)*

$$g_1(\mathbf{x}) = (k_1 + k_3)^2 - (1 + k_2)^2 < 0$$
$$g_2(\mathbf{x}) = (k_1 - k_3)^2 - (1 - k_2)^2 < 0$$

By introducing two slack-variables s_1 and s_2, the inequality constraints are converted into equality constraints, i.e.,

$$h_1(\mathbf{x}) = (k_1 + k_3)^2 - (1 + k_2)^2 + s_1^2 = 0 \tag{7.27a}$$

$$h_2(\mathbf{x}) = (k_1 - k_3)^2 - (1 - k_2)^2 + s_2^2 = 0 \tag{7.27b}$$

The design vector $\boldsymbol{\xi}$ thus becomes $\boldsymbol{\xi} = [k_1\ k_2\ k_3\ s_1\ s_2]^T$. From the initial guess $\boldsymbol{\xi}_0 = [0.28\ 0.74\ 0.12\ 1.69\ 1.2]^T$, the solution was found to be $\boldsymbol{\xi}_{\text{opt}} = [0.3248\ \ 0.5875$ $-0.009725\ 1.556\ 0.2415]^T$, and the corresponding link lengths are $a_1 \equiv 1$, $a_2 = 1.702$, $a_3 = 103.4$ and $a_4 = 102.8$. The Euclidean norm of the minimum design error is 5×10^{-2}.

The problem with this design is that it leads to a quite disproportionate linkage: two of its links have lengths two orders of magnitude bigger than those of the other two!

In the foregoing reference, a technique is introduced to eliminate this *dimensional unbalance* by means of a *penalty function*.

7.5.2 Penalty Functions

The idea behind penalty-function methods is to approach the optimum solution *asymptotically*, by extrapolation of a sequence of optimum solutions to unconstrained problems. There are two possibilities: the solution is approached either *within* the feasible region or from *without*, the penalty function being correspondingly referred to as *interior* or *exterior*. It is noteworthy that exterior penalty-function methods are applicable *only* to problems whereby the optimum finds itself at the boundary of the feasible region, but misses interior optima. Hence, we focus here on interior penalty functions.

Interior Penalty Functions

Given an objective function $f(\mathbf{x})$ subject to inequality constraints, as defined in eq.(7.1b), a sequence of interior penalty functions $\{\phi_k\}_1^\nu$ is constructed as

$$\phi_k(\mathbf{x}; r_k) \equiv f(\mathbf{x}) - r_k \sum_{i=1}^{p} \frac{1}{g_i(\mathbf{x})} \quad k = 1,\, 2,\, \cdots,\, \nu \tag{7.28}$$

where the term $-r_k \sum_{i=1}^{p}[1/g_i(\mathbf{x})]$ is called the *penalty term*, and all the r_k factors are positive and observe a decreasing order, i.e.,

$$r_1 > r_2 > r_3 \cdots > r_\nu > 0 \tag{7.29}$$

The idea here is that the search for the minimum is conducted within the feasible region. Under these conditions, the summation in the penalty term remains negative, and hence, a *positive* penalization is always *added* to the objective function. As the design-variable vector approaches the constraint $g_i(\mathbf{x}) = 0$, it does so *from the left*, i.e., $g_i(\mathbf{x}) \rightarrow 0^-$, and $1/g_i(\mathbf{x}) \rightarrow -\infty$, the penalty term thus becoming a "large" positive quantity, whose value is kept finite thanks to the presence of the "small" factor r_k.

Now, a sequence of unconstrained minimization problems is defined:

$$\phi_k(\mathbf{x}; r_k) \equiv f(\mathbf{x}) - r_k \sum_{i=1}^{p} \frac{1}{g_i(\mathbf{x})} \quad \rightarrow \quad \min_{\mathbf{x}}, \quad k = 1, 2, \cdots, \nu \tag{7.30}$$

Let $\mathbf{x}_o^1$, $\mathbf{x}_o^2$, ..., $\mathbf{x}_o^\nu$ be the sequence of corresponding unconstrained minima. Next, these minima are interpolated to a vector function $\mathbf{x}_o(r)$:

$$\mathbf{x}_o(r) \equiv \mathbf{c}_0 + \sum_{1}^{\nu-1} \mathbf{c}_k r^{k/2} \tag{7.31}$$

thereby obtaining a system of νn equations in νn unknowns, the n components of the ν unknown vector coefficients $\{\, \mathbf{c}_k \,\}_0^{\nu-1}$. Note that the foregoing equations are all linear in the unknowns, and hence, they can be solved for the unknowns using Gaussian elimination, as described below. First, eq.(7.31) is written for $r = r_i$, with $i = 1, 2, \ldots, \nu$:

$$\mathbf{x}_o(r_i) \equiv \mathbf{c}_0 + \mathbf{c}_1 r_i^{1/2} + \mathbf{c}_2 r_i^{2/2} + \cdots + \mathbf{c}_{\nu-1} r_i^{(\nu-1)/2} \tag{7.32}$$

or

$$\mathbf{x}_o(r_i) \equiv [\, \mathbf{c}_0 \quad \mathbf{c}_1 \quad \cdots \quad \mathbf{c}_{\nu-1} \,] \begin{bmatrix} 1 \\ r_i^{1/2} \\ \vdots \\ r_i^{(\nu-1)/2} \end{bmatrix}, \quad i = 1, 2, \ldots, \nu \tag{7.33}$$

In the next step, we regroup all ν vector equations above to produce a matrix equation. To this end, we define the matrices

$$\mathbf{R} \equiv \begin{bmatrix} 1 & 1 & \cdots & 1 \\ r_1^{1/2} & r_2^{1/2} & \cdots & r_\nu^{1/2} \\ \vdots & \vdots & \ddots & \vdots \\ r_1^{(\nu-1)/2} & r_2^{(\nu-1)/2} & \cdots & r_\nu^{(\nu-1)/2} \end{bmatrix} \tag{7.34a}$$

$$\mathbf{X}_o \equiv [\, \mathbf{x}_o(r_1) \quad \mathbf{x}_o(r_2) \quad \mathbf{x}_o(r_\nu) \,] \tag{7.34b}$$

$$\mathbf{C} \equiv [\, \mathbf{c}_0 \quad \mathbf{c}_1 \quad \cdots \quad \mathbf{c}_{\nu-1} \,] \tag{7.34c}$$

It is noteworthy that square matrices with the gestalt of $\mathbf{R}$ of eq.(7.34a) occur quite frequently in system theory, where they are termed *Vandermonde matrices*. For this reason, scientific code includes commands that ease the construction of such matrices. For example, Maple includes the command

$$\texttt{VandermondeMatrix}(\texttt{r}, \texttt{output})$$

in which `r` is the array $[r_1, r_2, \ldots, r_\nu]^T$, and `output` is the name assigned to the matrix thus constructed. The command includes various options.

Thus, the ν vector equations (7.32) become, in matrix form,

$$\mathbf{CR} = \mathbf{X}_o \tag{7.35a}$$

whence,

$$\mathbf{C} = \mathbf{X}_o\mathbf{R}^{-1} \tag{7.35b}$$

or, if eq.(7.35a) is written in the usual form, with the unknown matrix $\mathbf{C}$ to the left of its matrix coefficient, the foregoing equation should first be transposed, the result then being

$$\mathbf{C}^T = \mathbf{R}^{-T}\mathbf{X}_o^T \tag{7.35c}$$

with exponent $-T$ indicating the inverse of the transpose or, equivalently, the transpose of the inverse. Once the ν vector coefficients sought are available, the optimum of the inequality-constrained problem, $\mathbf{x}_{\text{opt}}$, is calculated as

$$\mathbf{x}_{\text{opt}} = \lim_{r\to 0} \mathbf{x}(r)$$

i.e.,

$$\mathbf{x}_{\text{opt}} = \mathbf{c}_0 \tag{7.36}$$

In computing the above value, note that $\mathbf{c}_0$ is the first column of the unknown matrix $\mathbf{C}$ or, equivalently, the first row of its transpose. In either case, it is not possible to obtain $\mathbf{c}_0$ as the solution of one single vector equation. A matrix equation must be solved in order to obtain $\mathbf{c}_0$. Such an equation is to be solved as a sequence of linear systems using LU-decomposition, one column of the matrix at a time.

Example 7.5.2 (A Two-dimensional Optimization Problem Subject to Inequality Constraints)

Consider an optimization problem with an objective function defined as

$$f = x^2 + 2y^2 \quad \to \quad \min_{x,y} \tag{7.37}$$

subject to the inequality constraints

$$g_1 \equiv -x \leq 0 \tag{7.38a}$$

$$g_2 \equiv -y \leq 0 \tag{7.38b}$$

$$g_3 \equiv 1 - x - y \leq 0 \tag{7.38c}$$

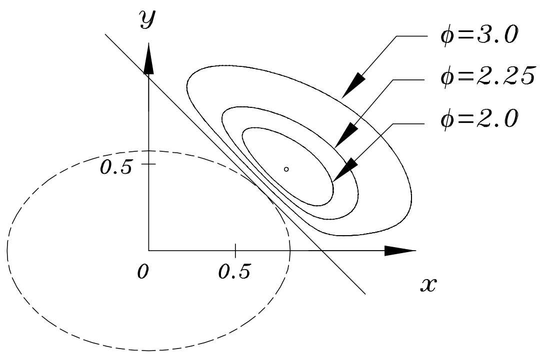


Figure 7.7: Isocontours of the penalty function with $r_1 = 0.1$

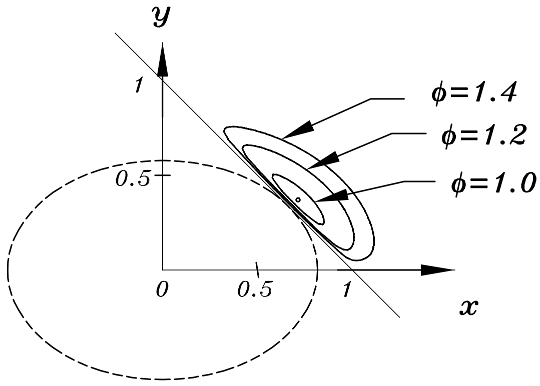


Figure 7.8: Isocontours of the penalty function with $r_2 = 0.01$

From a sketch of the constraints and the contours of the objective function in the x-y *plane, it should be apparent that the minimum of* f *is attained at a point where the gradient* ∇f *is parallel to the normal to the line* $g_3(x, y) = 0$*. The optimum values of* x *and* y *are, then*

$$x_{\text{opt}} = \frac{2}{3}, \quad y_{\text{opt}} = \frac{1}{3}$$

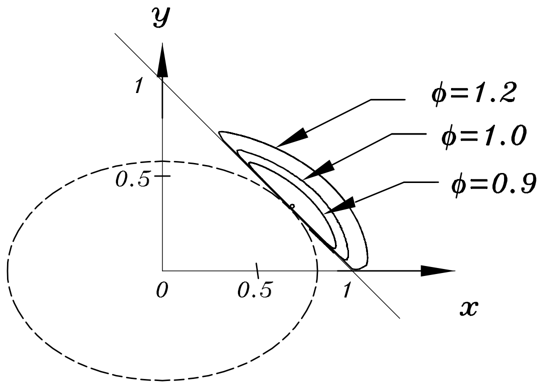


Figure 7.9: Isocontours of the penalty function with $r_3 = 0.001$

We demonstrate below the application of penalty functions to obtain the foregoing optimum. We have

$$\phi_k \equiv x^2 + 2y^2 - r_k\left(-\frac{1}{x} - \frac{1}{y} - \frac{1}{x+y-1}\right), \quad k = 1, \ldots, 3 \quad \rightarrow \quad \min_{x,y}$$

subject to no constraints, for

$$r_1 = 0.1, \quad r_2 = 0.01, \quad r_3 = 0.001$$

The penalty-function isocontours for different r_k values are shown in Figs. 7.7–7.9. In those figures, the isocontour of the objective function f that includes the constrained minimum is indicated with a dashed curve.

The optima $\mathbf{x}_o(r_k) \equiv [\, x_o(r_k),\, y_o(r_k)\,]^T$ for the three given values of r_k were found by the ODA, using subroutine ARBITRARY, *as*

$$\mathbf{x}_o(r_1) = \begin{bmatrix} 0.7941 \\ 0.4704 \end{bmatrix}, \quad \mathbf{x}_o(r_2) = \begin{bmatrix} 0.7140 \\ 0.3703 \end{bmatrix}, \quad \mathbf{x}_o(r_3) = \begin{bmatrix} 0.6836 \\ 0.3434 \end{bmatrix} \tag{7.39}$$

We now fit the values of $\{\mathbf{x}_o(r_k)\}_1^3$ to the function

$$\mathbf{x}_o(r) = \mathbf{c}_0 + \mathbf{c}_1 r^{1/2} + \mathbf{c}_2 r$$

We thus have

$$\mathbf{R} = \begin{bmatrix} 1 & 1 & 1 \\ 0.3163 & 0.1000 & 0.03163 \\ 0.1000 & 0.0100 & 0.00100 \end{bmatrix}, \quad \mathbf{X}_o = \begin{bmatrix} 0.7941 & 0.7140 & 0.6836 \\ 0.4704 & 0.3703 & 0.3434 \end{bmatrix} \tag{7.40}$$

The coefficient matrix $\mathbf{C}$ is thus found to be

$$\mathbf{C} = \mathbf{X}_o\mathbf{R}^{-1} = \begin{bmatrix} 0.6687 & 0.4790 & -0.2605 \\ 0.3317 & 0.3612 & 0.2443 \end{bmatrix} \tag{7.41}$$

Therefore,

$$\mathbf{x}_{\text{opt}} = \mathbf{c}_0 = \begin{bmatrix} 0.6687 \\ 0.3317 \end{bmatrix} \tag{7.42}$$

which yields the optimum with two significant digits of accuracy.

7.6 Direct Methods

Of the various direct methods for the solution of inequality-constrained problems, we shall discuss here three:

(i) The method of the feasible directions;

(ii) the generalized reduced-gradient method; and

(iii) the complex method.

7.6.1 The Method of the Feasible Directions

The method is due to Zoutendijk (1960). An outline of the method will be given in a future edition.

7.6.2 The Generalized Reduced-Gradient Method

This method, abbreviated as the GRG method, is an evolution of the *gradient-projection method* proposed by Rosen (1960). Further developments led to the *reduced-gradient method*, as applicable to arbitrary objective functions with linear equality constraints and inequalities of the form $\mathbf{x} \geq \mathbf{0}$. The generalization of the reduced-gradient method lies in its applicability to nonlinear equality and inequality constraints.

The method is best described if the problem at hand is formulated in a slightly different format than the one we have used so far: Given the objective function $f(\mathbf{x})$ and the C^1-continuous functions $h_j(\mathbf{x})$, for $j = 1, \ldots, l + p$,

$$f(\mathbf{x}) \quad \rightarrow \quad \min_{\mathbf{x}} \tag{7.43a}$$

subject to

$$h_j(\mathbf{x}) \leq 0, \quad j = 1, \ldots, p \tag{7.43b}$$

$$h_{p+j}(\mathbf{x}) = 0, \quad j = 1, \ldots, l \tag{7.43c}$$

$$l_i \leq x_i \leq u_i, \quad i = 1, \ldots, n \tag{7.43d}$$

The problem is first reformulated upon elimination of the first p inequalities by means of *nonnegative slack variables* $x_{n+1}, \ldots, x_{n+p}$. The constraints become now:

$$h_j(\mathbf{x}) + x_{n+j} = 0, \quad j = 1, \ldots, p \tag{7.44a}$$

$$h_{p+j}(\mathbf{x}) = 0, \quad j = 1, \ldots, l \tag{7.44b}$$

$$l_i \le x_i \le u_i, \quad i = 1, \ldots, n \tag{7.44c}$$

$$x_{n+j} \ge 0, \quad j = 1, \ldots, p \tag{7.44d}$$

Notice that the slack variables introduced in eq.(7.44a) are *non-negative*, while those of Subsection 7.5.1 are *quadratic*. The reason for the difference is that the latter were introdiced in the framework of *least squares*; the former have a historic origin, greatly influenced by the *simplex method* of *linear programming*, whereby all decision, or design, variables are regarded as non-negative.

We thus end up with a new problem:

$$f(\mathbf{x}) \quad \rightarrow \quad \min_{\mathbf{x}} \tag{7.45a}$$

subject to

$$h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, l + p \tag{7.45b}$$

$$l_i \le x_i \le u_i, \quad i = 1, \ldots, p + n \tag{7.45c}$$

Inequalities (7.45c) will be termed *bilateral*.

Remark: In light of eqs.(7.43c), the problem at hand has $n' \equiv (n - l)$ degrees of freedom, and hence, $(n - l)$ design variables—or a combination thereof—can be freely prescribed.

Strategy: *Partition* the set of design variables into two sets: $(n - l)$ *independent design variables* and $(l+p)$ *dependent design variables*, a.k.a. *state variables*. Relabel the design variables, if necessary.

Definitions:

$$\mathbf{x} \equiv \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix}, \quad \mathbf{y} \equiv \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-l} \end{bmatrix}, \quad \mathbf{z} \equiv \begin{bmatrix} x_{n-l+1} \\ x_{n-l+2} \\ \vdots \\ x_{p+n} \end{bmatrix}, \quad \mathbf{h} \equiv \begin{bmatrix} h_1(\mathbf{x}) \\ h_2(\mathbf{x}) \\ \vdots \\ h_{l+p} \end{bmatrix} \tag{7.46}$$

- $\mathbf{y}$: $(n - l)$-dimensional vector of independent variables
- $\mathbf{z}$: $(l + p)$-dimensional vector of dependent variables or state variables

- The partial derivatives $\partial f/\partial \mathbf{y}$, $\partial f/\partial \mathbf{z}$, $\partial \mathbf{h}/\partial \mathbf{y}$, and $\partial \mathbf{h}/\partial \mathbf{z}$ denote derivatives that **do not take into account** the dependence of $\mathbf{z}$ from $\mathbf{y}$
- $\mathrm{d}f/\mathrm{d}\mathbf{y}$ denotes the *total derivative* of f with respect to $\mathbf{y}$, which takes into account the dependent variables, and is, hence, a $(n-l)$-dimensional vector
- $\mathrm{d}\mathbf{h}/\mathrm{d}\mathbf{y}$ denotes the *total derivative* of $\mathbf{h}$ with respect to $\mathbf{y}$, which takes into account the dependent variables, and is, hence, a $(l+p)\times(n-l)$ matrix

That is,

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{y}} \equiv \begin{bmatrix} \mathrm{d}f/\mathrm{d}x_1 \\ \mathrm{d}f/\mathrm{d}x_2 \\ \vdots \\ \mathrm{d}f/\mathrm{d}x_{n-l} \end{bmatrix}, \quad \frac{\mathrm{d}\mathbf{h}}{\mathrm{d}\mathbf{y}} \equiv \begin{bmatrix} \frac{\mathrm{d}\mathbf{h}}{\mathrm{d}x_1} & \frac{\mathrm{d}\mathbf{h}}{\mathrm{d}x_2} & \cdots & \frac{\mathrm{d}\mathbf{h}}{\mathrm{d}x_{n-l}} \end{bmatrix} \tag{7.47}$$

Remark: Because of the constraints (7.45b), the partial derivatives with respect to $\mathbf{z}$ are dependent upon those with respect to $\mathbf{y}$. Indeed, since the equalities (7.45b) must hold, we must have

$$\mathbf{h}(\mathbf{y}, \mathbf{z}(\mathbf{y})) = \mathbf{0} \tag{7.48a}$$

which means that the total derivative of $\mathbf{h}$ with respect to $\mathbf{y}$ must vanish. From the *chain rule*,

$$\frac{\mathrm{d}\mathbf{h}}{\mathrm{d}\mathbf{y}} \equiv \frac{\partial \mathbf{h}}{\partial \mathbf{y}} + \frac{\partial \mathbf{h}}{\partial \mathbf{z}}\frac{\partial \mathbf{z}}{\partial \mathbf{y}} = \mathbf{0} \tag{7.48b}$$

Let

$$\mathbf{C} \equiv \frac{\partial \mathbf{h}}{\partial \mathbf{y}} = \begin{bmatrix} \partial h_1/\partial x_1 & \partial h_1/\partial x_2 & \cdots & \partial h_1/\partial x_{n-l} \\ \partial h_2/\partial x_1 & \partial h_2/\partial x_2 & \cdots & \partial h_2/\partial x_{n-l} \\ \vdots & \vdots & \ddots & \vdots \\ \partial h_{l+p}/\partial x_1 & \partial h_{l+p}/\partial x_2 & \cdots & \partial h_{l+p}/\partial x_{n-l} \end{bmatrix} \tag{7.48c}$$

$$\mathbf{D} \equiv \frac{\partial \mathbf{h}}{\partial \mathbf{z}} = \begin{bmatrix} \partial h_1/\partial z_1 & \partial h_1/\partial z_2 & \cdots & \partial h_1/\partial z_{l+p} \\ \partial h_2/\partial z_1 & \partial h_2/\partial z_2 & \cdots & \partial h_2/\partial z_{l+p} \\ \vdots & \vdots & \ddots & \vdots \\ \partial h_{l+p}/\partial z_1 & \partial h_{l+p}/\partial z_2 & \cdots & \partial h_{l+p}/\partial z_{l+p} \end{bmatrix} \tag{7.48d}$$

whence $\mathbf{C}$ is a $(l+p)\times(n-l)$ matrix, while $\mathbf{D}$ is a $(l+p)\times(l+p)$ matrix. We thus have

$$\Rightarrow \quad \mathbf{C} + \mathbf{D}\frac{\partial \mathbf{z}}{\partial \mathbf{y}} = \mathbf{0}$$

Solving for $\partial \mathbf{z}/\partial \mathbf{y}$ from the above equation yields

$$\frac{\partial \mathbf{z}}{\partial \mathbf{y}} = -\mathbf{D}^{-1}\mathbf{C} \tag{7.49}$$

Also notice that

$$\frac{\partial f}{\partial \mathbf{x}} \equiv \begin{bmatrix} \partial f / \partial \mathbf{y} \\ \partial f / \partial \mathbf{z} \end{bmatrix} \tag{7.50}$$

By application of the *chain rule*, the *total derivative* of f with respect to $\mathbf{y}$ is given by

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{y}} = \frac{\partial f}{\partial \mathbf{y}} + \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)^T \frac{\partial f}{\partial \mathbf{z}}$$

Substitution of eq.(7.49) into the above equation leads to

$$\begin{aligned} \frac{\mathrm{d}f}{\mathrm{d}\mathbf{y}} &= \frac{\partial f}{\partial \mathbf{y}} - (\mathbf{D}^{-1}\mathbf{C})^T \frac{\partial f}{\partial \mathbf{z}} \\ &= [\, \mathbf{1}_{n-l} \quad -(\mathbf{D}^{-1}\mathbf{C})^T \,] \begin{bmatrix} \partial f / \partial \mathbf{y} \\ \partial f / \partial \mathbf{z} \end{bmatrix} \equiv \mathbf{M}\frac{\partial f}{\partial \mathbf{x}} \end{aligned} \tag{7.51}$$

where

- $\mathbf{1}_{n-l}$: $(n-l) \times (n-l)$ identity matrix
- $\mathbf{M}$: a $(n-l) \times (p+n)$ matrix, namely,

$$\mathbf{M} \equiv [\, \mathbf{1}_{n-l} \quad -(\mathbf{D}^{-1}\mathbf{C})^T \,] \tag{7.52}$$

Definition: The *reduced gradient* of f is defined as the $(n-l)$-dimensional vector $\mathrm{d}f/\mathrm{d}\mathbf{y}$

Remarks:

- The reduced gradient of f is a linear transformation of the gradient of f with respect to $\mathbf{x}$, the transformation being given by matrix $\mathbf{M}$
- In the absence of inequalities (7.45c), the normality condition of Problem (7.45a & b) are

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{y}} \equiv \mathbf{M}\frac{\partial f}{\partial \mathbf{x}} = \mathbf{0} \tag{7.53}$$

  i.e., at a stationary point of Problem (7.45a & b), the gradient of f with respect to $\mathbf{x}$ need not vanish; only its *projection* onto matrix $\mathbf{M}$ must vanish.

Apart from inequalities (7.45c), which are relatively simple to handle, the problem can be treated as an *unconstrained* one.

It is noteworthy that $\mathbf{M}^T$ plays the role of the isotropic orthogonal complement $\mathbf{L}$ of $\mathbf{J} \equiv \partial \mathbf{h}/\partial \mathbf{x}$. Indeed, from eq.(7.49),

$$\Delta \mathbf{z} = \frac{\partial \mathbf{z}}{\partial \mathbf{y}}\Delta \mathbf{y} = -\mathbf{D}^{-1}\mathbf{C}\Delta \mathbf{y}$$

or

$$\mathbf{D}^{-1}\mathbf{C}\Delta\mathbf{y} + \Delta\mathbf{z} = \mathbf{0}_{l+p}$$

which can be cast in the form

$$[\,\mathbf{D}^{-1}\mathbf{C} \quad \mathbf{1}_{l'}\,]\begin{bmatrix}\Delta\mathbf{y}\\ \Delta\mathbf{z}\end{bmatrix} = \mathbf{0}_{l+p}$$

where $l' \equiv l + p$. Further, recalling the constraints $\mathbf{h}(\mathbf{x}) = \mathbf{0}$, we have

$$\frac{\partial\mathbf{h}}{\partial\mathbf{x}}\Delta\mathbf{x} \equiv \mathbf{J}\Delta\mathbf{x} = \mathbf{0}$$

whence,

$$\mathbf{J} = [\,\mathbf{D}^{-1}\mathbf{C} \quad \mathbf{1}_{l'}\,]$$

Now,

$$\mathbf{J}\mathbf{M}^T = [\,\mathbf{D}^{-1}\mathbf{C} \quad \mathbf{1}_{l'}\,]\begin{bmatrix}\mathbf{1}_{n-l}\\ -\mathbf{D}^{-1}\mathbf{C}\end{bmatrix} = \mathbf{D}^{-1}\mathbf{C} - \mathbf{D}^{-1}\mathbf{C} = \mathbf{O}_{l'\times n'}$$

with $n' \equiv n - l$. Therefore, $\mathbf{M}^T$ is an orthogonal complement of $\mathbf{J}$. However, contrary to $\mathbf{L}$, $\mathbf{M}^T$, or $\mathbf{M}$ for that matter, **is not isotropic**.

We need two search directions at the ith iteration: one for $\mathbf{y}$ and one for $\mathbf{z}$. To find these directions, we start by recalling eq.(7.48a), whence

$$\Delta\mathbf{h} = \underbrace{\frac{\partial\mathbf{h}}{\partial\mathbf{y}}}_{\equiv\mathbf{C}}\Delta\mathbf{y} + \underbrace{\frac{\partial\mathbf{h}}{\partial\mathbf{z}}}_{\equiv\mathbf{D}}\Delta\mathbf{z} = \mathbf{0}_{l+p} \tag{7.54}$$

Let, at the ith iteration,

$$\Delta\mathbf{y} = \lambda\mathbf{s}^i, \quad \Delta\mathbf{z} = \lambda\mathbf{t}^i, \quad \lambda > 0 \tag{7.55}$$

Substitution of eq.(7.55) into eq.(7.54) leads to

$$\lambda\mathbf{C}\mathbf{s}^i + \lambda\mathbf{D}\mathbf{t}^i = \mathbf{0}$$

Since $\lambda > 0$, the above equation leads, in turn, to

$$\mathbf{C}\mathbf{s}^i + \mathbf{D}\mathbf{t}^i = \mathbf{0}$$

whence,

$$\mathbf{t}^i = -\mathbf{D}^{-1}\mathbf{C}\mathbf{s}^i \tag{7.56}$$

All we need now is $\mathbf{s}^i$. Since we want to minimize $f(\mathbf{x},\,\mathbf{y})$, a plausible choice is

$$\mathbf{s}^i = -\left.\frac{\mathrm{d}f}{\mathrm{d}\mathbf{y}}\right|_{(\mathbf{y}^i,\mathbf{z}^i)} \tag{7.57}$$

Further, substitution of eq.(7.57) into eq.(7.56) leads to the desired expression for $\mathbf{t}^i$, namely,

$$\mathbf{t}^i = \mathbf{D}^{-1}\mathbf{C}\left.\frac{\mathrm{d}f}{\mathrm{d}\mathbf{y}}\right|_{(\mathbf{y}^i,\mathbf{z}^i)} \tag{7.58}$$

thereby obtaining the two desired search directions.

We thus have an update of $\mathbf{x}$:

$$\mathbf{x}^{i+1} \equiv \begin{bmatrix} \mathbf{y}^i + \lambda\mathbf{s}^i \\ \mathbf{z}^i + \lambda\mathbf{t}^i \end{bmatrix} \tag{7.59}$$

The optimum value λ^* of λ is found upon solving an unconstrained problem of one-dimensional minimization:

$$f(\mathbf{y}^i + \lambda\mathbf{s}^i,\, \mathbf{z}^i + \lambda\mathbf{t}^i) \quad \rightarrow \quad \min_{\lambda} \tag{7.60}$$

Remark: The foregoing optimization is implemented without consideration of inequalities (7.45c). Hence, a test must be conducted to verify whether those constraints are obeyed.

Adjustment: Let λ_i, for $i = 1, \ldots, p+n$, be the *positive* value of λ that renders one of the two inequalities of each of relations (7.45c) active. Then, let λ_{opt} be the adjusted value of λ that *does not violate* the above inequalities, i.e.,

$$\lambda_{\text{opt}} = \min\{\lambda^*,\, \{\,\lambda_i\,\}_1^{p+n}\} \tag{7.61}$$

Hence,

$$\mathbf{x}^{i+1} \equiv \begin{bmatrix} \mathbf{y}^i + \lambda_{\text{opt}}\mathbf{s}^i \\ \mathbf{z}^i + \lambda_{\text{opt}}\mathbf{t}^i \end{bmatrix} \tag{7.62}$$

Remark: After the foregoing adjustment has taken place, nothing guarantees that the constraints (7.45b) are verified. Hence, one further adjustment is needed:

Complying with the equality constraints: With $\mathbf{y}$ fixed to its current value, $\mathbf{y}^{\text{curr}}$, we correct the current value $\mathbf{z}^{\text{curr}}$ of $\mathbf{z}$ by means of the Newton-Raphson algorithm, i.e.,

1. $\mathbf{h}^{\text{curr}} \quad \rightarrow \quad \mathbf{h}(\mathbf{z}^{\text{curr}}; \mathbf{y}^{\text{curr}})$

2.

$$\mathbf{h}(\mathbf{z}^{\rm curr} + \Delta\mathbf{z}; \mathbf{y}^{\rm curr}) \approx \mathbf{h}^{\rm curr} + \left.\frac{\partial \mathbf{h}}{\partial \mathbf{z}}\right|_{\mathbf{y}=\mathbf{y}^{\rm curr},\, \mathbf{z}=\mathbf{z}^{\rm curr}} \Delta\mathbf{z} \quad \rightarrow \quad \mathbf{0}$$

3.

$$\Delta\mathbf{z} = -\left[\left.\frac{\partial \mathbf{h}}{\partial \mathbf{z}}\right|_{\mathbf{y}=\mathbf{y}^{\rm curr},\, \mathbf{z}=\mathbf{z}^{\rm curr}}\right]^{-1} \mathbf{h}^{\rm curr} \equiv -\mathbf{D}^{-1}\mathbf{h}^{\rm curr}$$

4. $\mathbf{z}^{\rm curr} \quad \rightarrow \quad \mathbf{z}^{\rm curr} + \Delta\mathbf{z}, \quad \mathbf{h}^{\rm curr} \rightarrow \quad \mathbf{h}(\mathbf{z}^{\rm curr}; \mathbf{y}^{\rm curr})$

5. `If` $\|\mathbf{h}^{\rm curr}\| \leq \epsilon$ `stop; else, go to 1`

where ϵ is a prescribed tolerance. Once the Newton-Raphson adjustment is completed, the optimization algorithm proceeds to the next iteration. The overall iterative procedure is finished when a convergence criterion has been met.

Drawbacks of the GRG method:

- Success is heavily dependent upon the user's choice of independent and dependent variables: Its rate of convergence dcereases as $\kappa(\mathbf{D})$ grows (Luenberger, 1984)
- The speed of convergence is slowed down by the Newton-Raphson iterations: we have an iteration loop within an exterior iteration loop!

In spite of the foregoing drawbacks, however, the GRG method is rather popular, for it is even available in Excel, an office-automation software package that has been used in the teaching of optimum design (Tai, 1998).

Example 7.6.1 The equilibrium configuration of a N-link chain (Luenberger, 1984)

We revisit here the problem of Example 6.4.2, for a N-link chain, as shown in Fig. 6.5, which we reproduce in Fig. 7.10 for quick reference.

We recall that the chain attains its equilibrium configuration when its potential energy attains its minimum value. As in the above-mentioned example, we use the configuration of Fig. 7.10b as initial guess *and find the equilibrium configuration for the values $N = 4$, $d = 1.5$ m, and $\ell = 0.5$ m*

Solution: In following Luenberger's formulation, we let the ith link span an x distance x_i and a y distance y_i. If $V \equiv \mu f(x_1, y_1, x_2, y_2 \ldots, x_N, y_N)$ denotes the

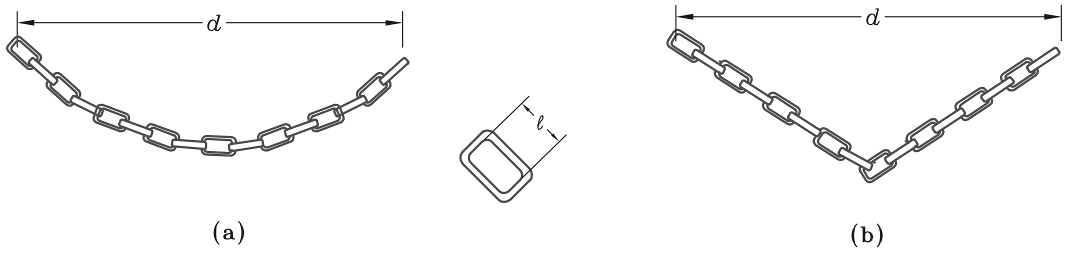


Figure 7.10: A N-link chain at: (a) its unknown equilibrium configuration; and (b) a known configuration to be used as an initial guess

potential energy of the chain, and μ is the mass density of the links per unit length, then minimizing V is equivalent to minimizing f, which is given by

$$\begin{aligned} f(x_1,\, y_1,\, x_2,\, y_2\, \ldots,\, x_N,\, y_N) &= \frac{1}{2}y_1 + (y_1 + \frac{1}{2}y_2) + \ldots \\ &\quad + (y_1 + y_2 + \ldots + y_{N-1} + \frac{1}{2}y_{N-1}) \\ &= \frac{1}{2}\sum_{i=1}^{N}[2(N-i)+1] \quad \rightarrow \quad \min_{\{x_i, y_i\}_1^N} \end{aligned}$$

subject to

$$\sum_{i=1}^{N} y_i = 0$$
$$x_i^2 + y_i^2 - 0.5^2 = 0$$
$$-0.5 \le x_i,\, y_i \le +0.5, \quad i = 1,\, \ldots,\, N$$

For $N = 4$, we have

$$f(x_1,\, y_1,\, x_2,\, y_2\, x_3,\, y_3,,\, x_4,\, y_4) = \frac{1}{2}(7y_1 + 5y_2 + 3y_3 + y_4) \quad \rightarrow \quad \min_{\{x_i, y_i\}_1^4}$$

subject to

$$\begin{aligned} h_1 &\equiv x_1 + x_2 + x_3 + x_4 - 1.5 = 0 \\ h_2 &\equiv y_1 + y_2 + y_3 + y_4 = 0 \\ h_3 &\equiv x_1^2 + y_1^2 - 0.5^2 = 0 \\ h_4 &\equiv x_2^2 + y_2^2 - 0.5^2 = 0 \\ h_5 &\equiv x_3^2 + y_3^2 - 0.5^2 = 0 \\ h_6 &\equiv x_4^2 + y_4^2 - 0.5^2 = 0 \end{aligned}$$
$$-0.5 \le x_i, \quad y_i \le +0.5, \quad i = 1,\, \ldots,\, 4$$

Use symmetry to simplify the problem:

$$x_3 = x_2, \quad x_4 = x_1, \quad y_3 = -y_2, \quad y_4 = -y_1$$

$$\Rightarrow f(x_1, y_1, x_2, y_2) \equiv 3y_1 + y_2 \quad \rightarrow \quad \min_{\{x_i, y_i\}_1^2}$$

Remark: Now constraint $h_2 = 0$ is identically verified and hence, is deleted. We are thus left with the constraints

$$h_1 \equiv x_1 + x_2 - 0.75 = 0$$
$$h_2 \equiv x_1^2 + y_1^2 - 0.5^2 = 0$$
$$h_3 \equiv x_2^2 + y_2^2 - 0.5^2 = 0$$
$$-0.5 \le x_i, \quad y_i \le +0.5, \quad i = 1, \ldots, 2$$

Now we have:

- Design variables: $x_1, y_1, x_2, y_2 \Rightarrow \quad n = 4$
- Equality constraints: $h_i = 0$, for $i = 1, 2, 3 \Rightarrow \quad l = 3$
- Degree of freedom: $n - l = 1 \Rightarrow$ one single independent variable. Choose y_1. Hence,

$$\mathbf{y} \equiv [y_1], \quad \mathbf{z} \equiv \begin{bmatrix} x_1 \\ x_2 \\ y_2 \end{bmatrix}$$

Preliminary calculations:

$$\frac{\partial f}{\partial \mathbf{y}} = \left[\frac{\partial f}{\partial y_1}\right] = [3], \quad \frac{\partial f}{\partial \mathbf{z}} = \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \\ \partial f/\partial y_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{h} \equiv \begin{bmatrix} x_1 + x_2 - 0.75 \\ x_1^2 + y_1^2 - 0.25 \\ x_2^2 + y_2^2 - 0.25 \end{bmatrix}$$

Hence,

$$\frac{\partial \mathbf{h}}{\partial \mathbf{y}} \equiv \left[\ \partial \mathbf{h}/\partial y_1 \quad \partial \mathbf{h}/\partial y_2\ \right] = \begin{bmatrix} 0 \\ 2y_1 \\ 0 \end{bmatrix} \equiv \mathbf{C},$$

$$\frac{\partial \mathbf{h}}{\partial \mathbf{z}} \equiv [\,\partial \mathbf{h}/\partial x_1 \quad \partial \mathbf{h}/\partial x_2 \quad \partial \mathbf{h}/\partial y_2\,] = \begin{bmatrix} 1 & 1 & 0 \\ 2x_1 & 0 & 0 \\ 0 & 2x_2 & 2y_2 \end{bmatrix} \equiv \mathbf{D}$$

Feasible initial guess: From Fig. 7.10b, in m,

$$x_1 = x_2 = 0.37500, \quad y_1 = y_2 = \frac{1}{2}\sqrt{1^2 - 0.75^2} = 0.33072 \quad \Rightarrow \quad \mathbf{x}^1 = \begin{bmatrix} 0.33072 \\ 0.37500 \\ 0.37500 \\ 0.33072 \end{bmatrix}$$

Hence,

$$f_{\text{curr}} \equiv f(\mathbf{x}^1) = 3 \times 0.33072 + 0.33072 = 1.32288$$
$$\mathbf{C} = \begin{bmatrix} 0 \\ 0.66144 \\ 0 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 1 & 1 & 0 \\ 0.75000 & 0 & 0 \\ 0 & 0.7500 & 0.66144 \end{bmatrix}$$

and

$$\mathbf{D}^{-1} = \begin{bmatrix} 0 & 1.3333 & 0 \\ 1 & -1.3333 & 0 \\ -1.1339 & 1.5119 & 1.5119 \end{bmatrix}$$

Further,

$$\frac{\partial \mathbf{z}}{\partial \mathbf{y}} = -\mathbf{D}^{-1}\mathbf{C} \tag{7.63}$$
$$= -\begin{bmatrix} 0 & 1.3333 & 0 \\ 1 & -1.3333 & 0 \\ -1.1339 & 1.5119 & 1.5119 \end{bmatrix} \begin{bmatrix} 0 \\ 0.66144 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.88192 \\ 0.88192 \\ -1.00000 \end{bmatrix} \tag{7.64}$$

$$\Rightarrow \quad \frac{\mathrm{d}f}{\mathrm{d}\mathbf{y}} = \frac{\partial f}{\partial \mathbf{y}} + \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)^T \frac{\partial f}{\partial \mathbf{z}} = \frac{\partial f}{\partial \mathbf{y}} - (\mathbf{D}^{-1}\mathbf{C})^T \frac{\partial f}{\partial \mathbf{z}}$$
$$= 3 + [\,-0.88192 \quad 0.88192 \quad -1.0000\,] \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = [3] - [1] = [2]$$

Search directions:

$$\mathbf{s}^1 = -\left.\frac{\mathrm{d}f}{\mathrm{d}\mathbf{y}}\right|_{\mathbf{x}=\mathbf{x}^1} = -[2]$$

Hence,

$$\mathbf{t}^1 = -\mathbf{D}^{-1}\mathbf{C}\mathbf{s}^1 = \begin{bmatrix} -0.88192 \\ 0.88192 \\ -1.00000 \end{bmatrix} [-2] = \begin{bmatrix} 1.7638 \\ -1.7638 \\ 2.0000 \end{bmatrix}$$

$$\Rightarrow \quad \Delta\mathbf{x} = \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix} = \begin{bmatrix} \lambda\mathbf{s}^1 \\ \lambda\mathbf{t}^1 \end{bmatrix} = \lambda \begin{bmatrix} -2.0000 \\ 1.7638 \\ -1.7638 \\ 2.0000 \end{bmatrix}$$

and

$$\mathbf{x}^{\text{new}} = \mathbf{x}^{\text{curr}} + \Delta\mathbf{x} = \begin{bmatrix} 0.33072 - 2.0000\lambda \\ 0.37500 + 1.7638\lambda \\ 0.37500 - 1.7638\lambda \\ 0.33072 + 2.0000\lambda \end{bmatrix}$$

Thus,

$$\begin{aligned} f(\mathbf{x} + \Delta\mathbf{x}) = f(\lambda) &= 3(y_1 + \Delta y_1) + (y_2 + \Delta y_2) \\ &= 3(0.33072 - 2\lambda) + 0.33072 + 2\lambda = 1.3229 - 4\lambda \end{aligned}$$

Hence, in order to minimize $f(\lambda)$ we must make λ (> 0) *as large as the constraints allow us.* In the next step we find the set $\{\,\lambda_i\,\}_1^4$ allowing us to attain the equality on one side of the set of inequalities (7.45c):

$$\begin{aligned} y_1: &\quad 0.33072 - 2.0000\lambda_1 = -0.5 \quad \Rightarrow \quad \lambda_1 = \frac{0.83072}{2.0000} = 0.41536 \\ x_1: &\quad 0.37500 + 1.7638\lambda_2 = 0.5 \quad \Rightarrow \quad \lambda_2 = \frac{0.12500}{1.7638} = 0.070868 \\ x_2: &\quad 0.37500 - 1.7638\lambda_2 = -0.5 \quad \Rightarrow \quad \lambda_3 = \frac{0.87500}{1.7638} = 0.49608 \\ y_2: &\quad 0.33072 + 2.0000\lambda_1 = 0.5 \quad \Rightarrow \quad \lambda_4 = \frac{0.16928}{2.0000} = 0.084640 \end{aligned}$$

whence, $\lambda_{\text{opt}} = \min\{0.41536,\ 0.070868,\ 0.49608,\ 0.084640\} = 0.070868$
Therefore,

$$\mathbf{x}_{\text{new}} = \begin{bmatrix} 0.33072 - 2.0000 \times 0.070868 \\ 0.37500 + 1.7638 \times 0.070868 \\ 0.37500 - 1.7638 \times 0.070868 \\ 0.33072 + 2.0000 \times 0.070868 \end{bmatrix} = \begin{bmatrix} 0.18898 \\ 0.5000 \\ 0.2500 \\ 0.47246 \end{bmatrix}$$

$$\Rightarrow \quad f_{\text{new}} = 3 \times 0.18898 + 0.47246 = 1.0394$$

which means that we brought down the objective function by 21% of its original value in one single iteration. However, nothing guarantees that the equality constraints are satisfied. Let us verify:

$$\mathbf{h}^{\text{new}} = \mathbf{h}(\mathbf{x}^{\text{new}}) = \begin{bmatrix} 0.5000 + 0.25000 - 0.75000 \\ 0.5000^2 + 0.18898^2 - 0.5000^2 \\ 0.25000^2 + 0.47246^2 - 0.5000^2 \end{bmatrix} = \begin{bmatrix} 0.0000 \\ 0.035713 \\ 0.035718 \end{bmatrix} \neq \mathbf{0}$$

Hence, a correction to $\mathbf{z}$, with $\mathbf{y}$ kept at its current value $\mathbf{y}^{\text{curr}}$, is warranted:

$$\mathbf{D}_{\text{new}}\Delta\mathbf{z} = -\mathbf{h}^{\text{new}}$$

where

$$\mathbf{D}_{\rm new} \equiv \mathbf{D}(\mathbf{x}^{\rm new}) = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0.5000 & 0.94492 \end{bmatrix}$$

Hence,

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0.5000 & 0.94492 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta y_2 \end{bmatrix} = \begin{bmatrix} 0.0000 \\ 0.035713 \\ 0.035718 \end{bmatrix} \quad \Rightarrow \quad \Delta\mathbf{z} = \begin{bmatrix} -0.035713 \\ 0.035713 \\ -0.056698 \end{bmatrix}$$

Thus,

$$\mathbf{h}(\mathbf{z}^{\rm new} + \Delta\mathbf{z}; \mathbf{y}^{\rm new}) = \begin{bmatrix} 0.0000 \\ 0.0012786 \\ 0.0044866 \end{bmatrix}$$

Hence, the norm of $\mathbf{h}$ has been brought down by one order of magnitude. Besides, $f(\mathbf{z}^{\rm new} + \Delta\mathbf{z}; \mathbf{y}^{\rm new}) = 0.9827$, and hence, the objective function was brought down by an additional 5.5%, thereby completing one full iteration. Further iterations are left to the reader as an excercise.

7.6.3 The Complex Method

A *complex* in $\mathbb{R}^n$ is a polyhedron with $m > n+1$ vertices; e.g., in $2D$, a complex is a quadrilateron; in $3D$ a cube is an example of a complex. We describe below a method due to Box (1965). The method is based on a complex $\mathcal{C}$ with $m = 2n$ vertices, for $n \geq 2$. Implementations are reported in (Kuester and Mize, 1973) and (Xu et al., 1994).

Box's Algorithm

1. `Given one feasible vertex` $\mathbf{x}^1$ `of the initial complex, generate the remaining` $2n-1$ `vertices so that complex is feasible:` $\mathcal{C} = \{\mathbf{x}^i\}_1^{2n} \in \mathcal{R}_F$`, the feasible region`

2. `Let` $f_i \equiv f(\mathbf{x}^i)$ `and` $f_M = \max\{f_i\}^{2n}$ $\%\mathbf{x}^M$ `is the worst vertex`

3. `Let` $\mathcal{C}' = \{\mathbf{x}^i\}_{i=1,\ i\neq M}^{2n}$ `and let` $\overline{\mathbf{x}}$ `be the position vector of the centroid of` $\mathcal{C}'$`, i.e.,`

$$\overline{\mathbf{x}} = \frac{1}{2n-1}\left(\sum_{i=1}^{2n} \mathbf{x}^i - \mathbf{x}^M\right)$$

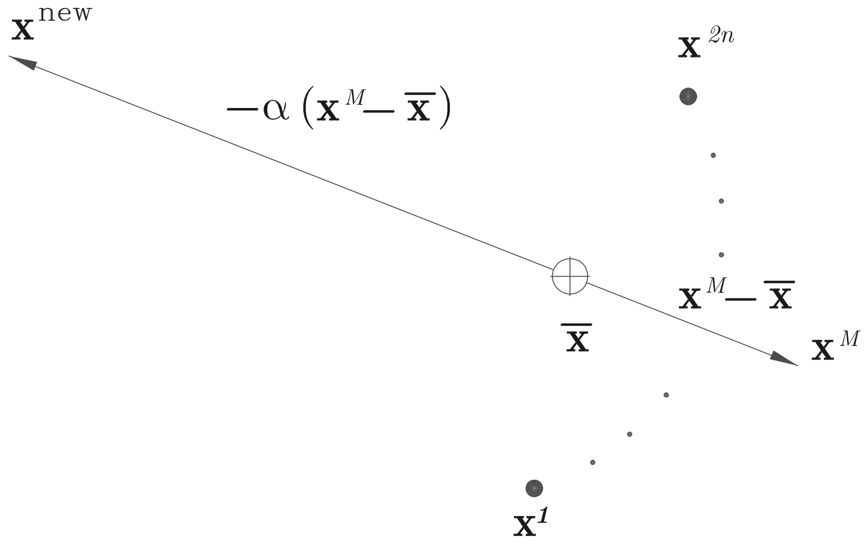


Figure 7.11: Replacement of the worst vertex of the complex by a reflection

4. `Recover lost vertex of complex by reflecting` $\mathbf{x}^M$ `about` $\overline{\mathbf{x}}$ `by means of`

$$\mathbf{x}^{\text{new}} \leftarrow \mathbf{x}^M \leftarrow \overline{\mathbf{x}} - \alpha(\mathbf{x}^M - \overline{\mathbf{x}}) \equiv (1+\alpha)\overline{\mathbf{x}} - \alpha\mathbf{x}^M, \ \alpha > 0 \quad (\alpha_{\text{Box}} = 1.3)$$

5. `if` $\mathbf{x}^{\text{new}} \in \mathcal{R}_F$ `, continue; else`

   5.1 $\mathbf{x}^{\text{new}} \leftarrow \frac{1}{2}(\overline{\mathbf{x}} + \mathbf{x}^{\text{new}})$

   5.2 `if` $\mathbf{x}^{\text{new}} \in \mathcal{R}_F$ `, continue; else go to 5.1`

   `abort if too many iterations`

6. `go to 2; if` $\mathbf{x}^{\text{new}}$ `is not new worst vertex continue; else`

$$\mathbf{x}^{\text{new}} \leftarrow \beta(\mathbf{x}^M - \overline{\mathbf{x}}), \ 0 < \beta < 1$$

7. `stop when convergence criterion has been met.`

A possible convergence criterion is to stop when the difference between the maximum value f_M of the objective function and its minimum, f_m, is smaller than a prescribed ratio ϵ_1 times the same difference at the original complex. An alternative criterion involves the *size* of the current complex, given by the rms value of the distances of the vertices from the centroid: whenever this values is smaller than a prescribed ratio ϵ_2 times the corresponding value for the original complex, the procedure stops. A combination of the two criteria is advisable.

More on the Chain Problem

Example 7.6.2 *Find* $\mathbf{f}$ *that balances the weight of the links, noting that* $\mathbf{f}$ *is horizontal, for an arbitrary configuration* $\begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix}^T$.

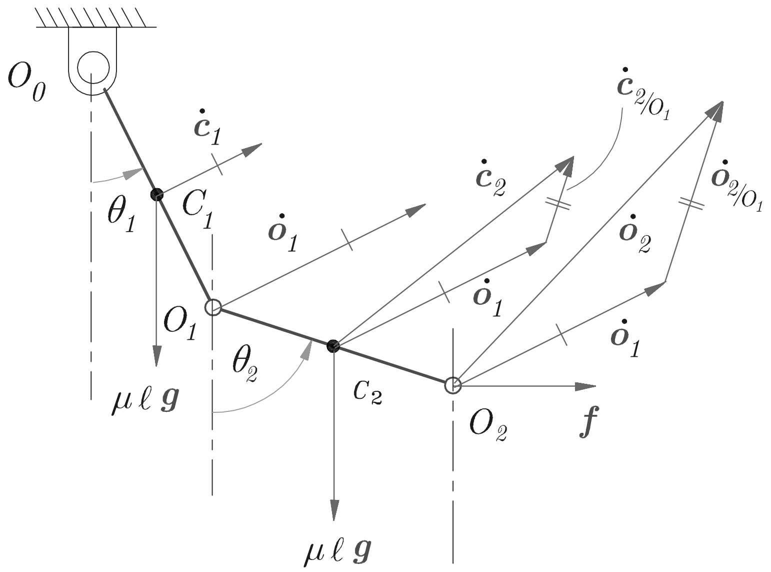


Figure 7.12: Chain

Solution: Apply the Principle of Virtual Work: $\sum \pi_i = 0$, where π_i is power developed by the ith external force. According to Fig.7.12, π_1 and π_2 is power developed by weight of 1st and 2nd link, while π_3 is power developed by force $\mathbf{f}$. Hence, we have

$$\pi_1 = \mu\ell\mathbf{g}\cdot\dot{\mathbf{c}}_1, \quad \pi_2 = \mu\ell\mathbf{g}\cdot\dot{\mathbf{c}}_2, \quad \pi_3 = \mathbf{f}\cdot\dot{\mathbf{o}}_2$$

Then, we have

$$\mathbf{g}\cdot\dot{\mathbf{c}}_1 = -g\frac{\ell}{2}\dot{\theta}_1\sin\theta_1$$

$$\mathbf{g}\cdot\dot{\mathbf{c}}_2 = \mathbf{g}\cdot(\dot{\mathbf{o}}_1 + \dot{\mathbf{c}}_{2/o_1}) = 2\mathbf{g}\cdot\dot{\mathbf{c}}_1 + \mathbf{g}\cdot\dot{\mathbf{c}}_{2/o_1} = -g\ell(\dot{\theta}_1\sin\theta_1 + \frac{1}{2}\dot{\theta}_2\sin\theta_2)$$

$$\mathbf{f}\cdot\dot{\mathbf{o}}_2 = \mathbf{f}\cdot(\dot{\mathbf{o}}_1 + 2\dot{\mathbf{o}}_{2/o_1}) = F\ell(\dot{\theta}_1\cos\theta_1 + \dot{\theta}_2\cos\theta_2)$$

where $F \equiv \|\mathbf{f}\|$. Applying the Principle of Virtual Work, we derive

$$-\mu g \frac{\ell^2}{2} \dot{\theta}_1 \sin\theta_1 - \mu g \ell^2 (\dot{\theta}_1 \sin\theta_1 + \frac{1}{2}\dot{\theta}_2 \sin\theta_2) + F\ell(\dot{\theta}_1 \cos\theta_1 + \dot{\theta}_2 \cos\theta_2) = 0$$

or

$$[-\mu g \ell (\frac{1}{2}\sin\theta_1 + \sin\theta_1) + F\cos\theta_1]\dot{\theta}_1 + [-\frac{1}{2}\mu g \ell \sin\theta_2 + F\cos\theta_2]\dot{\theta}_2 = 0$$

Since $\dot{\theta}_1$ and $\dot{\theta}_2$ are independent, we obtain

$$\begin{aligned} -\mu g \ell \frac{3}{2} \sin\theta_1 + F\cos\theta_1 &= 0, \\ -\mu g \ell \frac{1}{2} \sin\theta_2 + F\cos\theta_2 &= 0, \end{aligned}$$

or

$$3\sin\theta_1 - \lambda\cos\theta_1 = 0, \quad \sin\theta_2 - \lambda\cos\theta_2 = 0 \tag{7.65}$$

where we introduce the notation

$$\lambda \equiv \frac{F}{\mu g \ell/2}$$

Equations (7.65) represent first-order normality conditions sought.

Bibliography

Angeles, J. and C. S. López-Cajún. 1991. *Optimization of Cam Mechanisms*. Dordrecht: Kluwer Academic Publishers.

Bertsekas, D. P. 1995. *Nonlinear Programming*. Belmont, Massachusetts: Athena Scientific.

Box, M. J. 1965. A new method of constrained optimization and a comparison with other methods. *Computer Journal* 8:42–52.

Boyd, S. and L. Vandenberghe. 2004. *Convex Optimization*. Cambridge: Cambridge University Press.

Brent, R. P. 1972. *Algorithms for Minimization Without Derivatives*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Broyden, C. G. 1970. The convergence of a class of double-rank minimization algorithm. *J. Inst. Maths. Applications* 6:76–90.

Culioli, J.-C. 1994. *Introduction a l'optimisation*. Paris: Ellipses Publisher.

Dahlquist, G. and Å. Björck. 1974. *Numerical methods*. Englewood Cliffs, N. J.: Prentice-Hall,.

Fletcher, R. 1970. A new approach to variable metric algorithms. *Computer Journal* 13:317–322.

French, M. E. 1992. *Form, Structure and Mechanism*. London: Macmillan.

Gleick, J. 1988. *Chaos: making a new science*. New York, N.Y.,: Penguin.

Goldfarb, D. 1970. A family of variable metric updates derived by variational means. *Mathematics of Computing* 24:23–26.

Kuester, J. L. and J. H. Mize. 1973. *Optimization Techniques with Fortran*. McGraw-Hill Book Co.

Kuhn, H. W. and A. W. Tucker. 1951. Nonlinear programming. In J. Neyman (ed.), *Second Berkeley Simposium*, 481–492, Berkeley: University of California Press.

Lalee, M., J. Nocedal, and Plantenga. 1998. On the implementation of an algorithm for large-scale equality constrained optimization. *SIAM J. Optim.* 8:682–706.

Liu, Z. and J. Angeles. 1992. Least-square optimization of planar and spherical four-bar function generator under mobility constraint. *ASME, J. of Mech. Des* 114:569–573.

Livio, M. 2002. *The Golden Ratio.The Story of Phi, the World's Most Astonishing Number*. New York: Broadway Books.

Luenberger, D. G. 1984. *Linear and Nonlinear Programming*. Reading (MA): Addison-Wesley Publishing Company, 2nd ed.

Mandelbrot, B. B. 1983. *The Fractal Geometry of Nature*. NY: W. H. Freeman and Company, 3rd ed.

Murray, W. 1997. Sequential quadratic programming methods for large problems. *Comput. Optim. Appl* 7:127–142.

Nelder, J. A. and R. Mead. 1965. A simplex method for function minimization. *Computer Journal* 7:308.

Powell, M. J. D. 1969. *Optmization*, chap. A method for Nonlinear Constraints in Minimizing Problems. Academic Press, 283–298.

Rao, S. S. 1996. *Engineering Optimization*. New York: John Wiley and Sons, Inc., 3rd ed.

Rosen, J. B. 1960. The gradient projection method for nonlinear programming, Part I, Linear constraints. *Journal of SIAM* 8.

Salisbury, J. K. and J. J. Craig. 1982. Articulated hands: Force and kinematic issues. *The Int. J. Robotics Res.* 1(2):3.

Shanno, D. F. 1970. Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computing* 24:647–656.

Shigley, J. and C. Mischke. 1989. *Mechanical Engineering Design*. McGraw-Hill, Inc., New York.

Spendley, W., G. R. Hext, and F. R. Himsworth. 1962. Sequential application of simplex designs in optimization and evolutionary operation. *Technometrics* 4:441.

Taguchi, G. 1988. The development of quality engineering. *The ASI Journal* 1(1).

Tai, K. 1998. Integrated design optimization and analysis using a spreadsheet application. *Int. Journal of Mechanical Engineering Education* 27(1):29–40.

Teng, C. P. and J. Angeles. 2001. A sequential-quadratic-programming algorithm using orthogonal decomposition with Gerschgorin stabilization. *ASME J. of Mechanical Design* 123:501–509.

Varga, R. S. 2000. *Matrix Iterative Analysis*. Englewood Cliffs, N. J.: Prentice-Hall Inc.

Wilde, D. 1992. Monotonicity analysis of Taguchi's robust circuit design problem. *ASME Journal of Mechanical Design* 114:616–619.

Xu, P., T. Ramden, R. T. Burton, P. Krus, and C. Sargent. 1994. Feasibility of training a neural network based hydraulic component simulator using the comlpex method. In *Seventh Bath International Fluid Power Workshop*.

Zoutendijk, G. 1960. *Methods of Feasible Directions*. Amsterdam: Elsevier Publishing Company.

Index