The Solution of Nonlinear Programming Problems with the Orthogonal Decomposition Algorithm—A Tutorial

Chin-Pun Teng and Jorge Angeles

September 2000

McGill Research Centre for Intelligent Machines &
Department of Mechanical Engineering
McGill University

Contents

Abstract

The aim of this tutorial is to outline the application of the orthogonal decomposition algorithm to nonlinear programming problems. To this end, applications of this algorithm to linear-quadratic programming problems are first illustrated, based on which, the nonlinear problems are solved iteratively. Further applications to problems with arbitrary objective functions are also covered.

The use of the software package *ODA*, implementing the above algorithm, is also outlined here. Examples for each type of problem are given.

1 Introduction

Many engineering design problems are formulated as optimization problems, the design optimization problem being well established in the context of mathematical programming (Mangasarian, 1972; Seireg, 1972; Fox and Gupta, 1973). This problem consists in the minimization of a real scalar objective function of a set of design variables subject to several constraints. Usually, constraints for optimization problems consist of both equalities and inequalities. However, inequalities may be eliminated or transformed into equality constraints by properly modifying the problem formulation. The main item introduced in this tutorial is the *orthogonal decomposition algorithm* (ODA), which handles 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*.

Implementation of the ODA in a software package, called QUADMIN, was developed by Angeles et al. (1990) for linear and nonlinear equality-constrained least-square optimization. Further applications of the ODA in combination with Gerschgorin stabilization for more general applications, in the framework of sequential quadratic programming (SQP), discussed in this tutorial as well, were developed by Teng and Angeles (1999).

Sequential quadratic programming appears to be the most commonly used and reliable method in nonlinear programming. SQP is well suited for the solution of nonlinear programming problems, as reported by Murray(1997) and Lalee, Nocedal and Plantenga (1998). In SQP, the nonlinear optimization problem is approximated by a sequence of quadratic programs (QP). At each iteration, if the *Hessian matrix* of the objective function in the sub-QP problem is not positive-definite, the Hessian matrix is modified to render it positive-definite, a procedure called *Hessian-stabilization*. Then, the sub-QP is solved by means of any algorithm suitable for QP problems. The procedure is terminated as 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, Fletcher, Goldfarb, 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).

In this tutorial, a method of Hessian stabilization is described that is based on the Gerschgorin Theorem (Varga, 1962). Besides, the orthogonal-decomposition algorithm (ODA) is applied in solving the sequential QP. The software package, ODA, implementing this method and the others discussed in the tutorial, is also outlined.

2 Nomenclature

1: the $k \times k$ identity matrix, where k is an integer.

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

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

C: $l \times n$ coefficient matrix of the linear system $\mathbf{Cx} = \mathbf{d}$.

d: l-dimensional vector of the underdetermined linear system $\mathbf{Cx} = \mathbf{d}$, with $l < q - n$.

f: the objective function.

$\mathbf{g}(\mathbf{x})$: m-dimensional nonlinear vector function of the design-variable vector $\mathbf{x}$, representing the left-hand side of the inequality constraints: $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$

G: $m \times n$ Jacobian matrix of the vector function $\mathbf{g}(\mathbf{x})$: $\mathbf{G} \equiv \partial\mathbf{g}/\partial\mathbf{x}$

$\mathbf{h}(\mathbf{x})$: l-dimensional nonlinear vector function of the design-variable vector $\mathbf{x}$, representing the left-hand side of the equality constraints: $\mathbf{h}(\mathbf{x}) = \mathbf{0}$

$\mathbf{H}_i$: the ith Householder reflection.

$\mathbf{J}$: $l \times n$ gradient matrix of the vector function $\mathbf{h}(\mathbf{x})$: $\mathbf{J} \equiv \partial\mathbf{h}/\partial\mathbf{x}$.

l: number of constraint equations $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ or $\mathbf{Cx} = \mathbf{d}$.

$\mathbf{L}$: the *isotropic* orthogonal complement of $\mathbf{C}$ or, alternatively, of $\mathbf{J}$.

$\mathbf{M}^I$: the *left* Moore-Penrose generalized inverse of the *full-rank* $q \times n$ matrix $\mathbf{M}$, with $q > n$: $\mathbf{M}^I \equiv (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$.

n: number of design variables; the dimension of the design vector $\mathbf{x}$.

$\mathbf{N}^\dagger$: the *right* Moore-Penrose generalized inverse of the *full-rank* $q \times n$ matrix $\mathbf{N}$, with $q < n$: $\mathbf{N}^\dagger \equiv \mathbf{N}^T(\mathbf{N}\mathbf{N}^T)^{-1}$.

$\mathbf{O}$: the $k \times l$ zero matrix, for given integers k and l.

q: number of equations $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{0}$ or $\mathbf{Ax} = \mathbf{b}$.

$\mathbf{U}$: an upper-triangular matrix.

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

$\mathbf{W}$: an $q \times q$ symmetric and positive-semidefinite weighting matrix.

$\mathbf{x}$: n-dimensional design vector; its components are the design variables.

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

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

$\boldsymbol{\phi}$: q-dimensional nonlinear vector function $\boldsymbol{\phi} = \boldsymbol{\phi}(\mathbf{x})$.

$\boldsymbol{\Phi}$: $q \times n$ gradient matrix of the vector function $\boldsymbol{\phi}(\mathbf{x})$: $\boldsymbol{\Phi} \equiv \partial\boldsymbol{\phi}/\partial\mathbf{x}$.

$\|\cdot\|$: Euclidean norm of its vector argument $(\,\cdot\,)$.

3 Unconstrained Linear Least-Square Problems

3.1 The Least-Square Solution of Overdetermined Linear Systems

An overdetermined system of linear equations is defined as

$$\mathbf{Ax} = \mathbf{b} \tag{1}$$

where $\mathbf{A}$ is a $q \times n$ matrix, with $q > n$. For such a system, there are no solutions in general, but, if $\mathbf{A}$ has full rank, then a unique approximate solution, denoted as $\mathbf{x}_L$, can be found, which verifies eq.(1) with the minimum least-square error. Hence, the said solution is termed the *least-square solution* of that equation. The error $\mathbf{e}$

incurred in approximating eq.(1) is defined as

$$\mathbf{e} \equiv \mathbf{b} - \mathbf{A}\mathbf{x} \tag{2}$$

Thus, the least-square solution $\mathbf{x}_L$ minimizes

$$\|\mathbf{e}\|^2 \equiv \|\mathbf{b} - \mathbf{A}\mathbf{x}\|^2 \equiv (\mathbf{b} - \mathbf{A}\mathbf{x})^T(\mathbf{b} - \mathbf{A}\mathbf{x}) \tag{3}$$

It can be shown (Strang, 1988) that the vector $\mathbf{x}$ that minimizes the Euclidean norm of $\mathbf{e}$ verifies the system of equations

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b} \tag{4}$$

which is the *normality condition* of the least-square problem at hand, also known as the *normal equations* of the problem. If $\mathbf{A}$ is of full-rank, then the least-square solution can be expressed as

$$\mathbf{x}_L = \mathbf{A}^I\mathbf{b} \tag{5a}$$

where

$$\mathbf{A}^I \equiv (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T \tag{5b}$$

which is called the *left Moore-Penrose generalized inverse* of the rectangular matrix $\mathbf{A}$.

Prior to continuing the discussion of the problem at hand, we need some background, which is given below:

Let $\delta\mathbf{A}$ and $\delta\mathbf{b}$ be the roundoff-error vectors in $\mathbf{A}$ and $\mathbf{b}$, respectively which stem from various sources. For example, the entries of $\mathbf{A}$ and $\mathbf{b}$ are usually obtained as the output of a computational algorithm; in some instances, these entries are the results of measurements, and hence, are corrupted with instrument noise. The roundoff-error vector incurred in solving eq.(1) is denoted by $\delta\mathbf{x}$. The relative roundoff errors in the computed solution as well as in the data, $\epsilon_{\mathbf{x}}$, $\epsilon_{\mathbf{A}}$ and $\epsilon_{\mathbf{b}}$, are defined as

$$\epsilon_{\mathbf{x}} \equiv \frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|}, \quad \epsilon_{\mathbf{A}} \equiv \frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|}, \quad \epsilon_{\mathbf{b}} \equiv \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} \tag{6}$$

where $\|\cdot\|$ denotes a *vector norm* or a *matrix norm* of vector $(\cdot)$ or, correspondingly, of matrix $(\cdot)$. Now, a norm is the generalized concept of absolute value, as applicable to arrays. For example, the *Chebyshev norm* of a vector is the largest absolute value of the components of the vector, while the *Euclidean norm* of the same is the well-known *magnitude* of this array, i.e., the square root of the sum of the squares of the vector components. The counterpart of the vector Chebyshev norm is the *matrix*

Chebyshev norm, defined as the largest absolute value of the entries of the matrix, while that of the vector Euclidean norm is **not** the Euclidean matrix norm, but rather the *Frobenius norm*, defined as the square root of the sum of the squares of the entries of the matrix. That is, if $\|\mathbf{A}\|_E$ and $\|\mathbf{A}\|_F$ denote the Euclidean and the Frobenius norms of the $q \times n$ matrix $\mathbf{A}$, then

$$\|\mathbf{A}\|_E = \max_i \{\sqrt{\lambda_i}\}_1^n, \quad \|\mathbf{A}\|_F = \sqrt{\mathrm{tr}(\mathbf{A}\mathbf{W}\mathbf{A}^T)} \tag{7}$$

where $\{\lambda_i\}_1^n$ is the set of eigenvalues of $\mathbf{A}^T\mathbf{A}$; the square roots are the *singular values of* $\mathbf{A}$. Moreover, $\mathbf{W}$ is a positive-definite weighting matrix, introduced for normalization purposes. For example, if $\mathbf{W} = (1/n)\mathbf{1}$, with $\mathbf{1}$ defined as the $n \times n$ identity matrix, them the Frobenius norm of $\mathbf{1}$ is exactly unity.

The relative roundoff errors 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)

$$\epsilon_{\mathbf{x}} \leq \kappa(\mathbf{A})(\epsilon_{\mathbf{A}} + \epsilon_{\mathbf{b}}) \tag{8}$$

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

$$\kappa(\mathbf{A}) \equiv \|\mathbf{A}\|\|\mathbf{A}^{-1}\| \tag{9}$$

for*any* matrix norm. The condition number, moreover, is bounded from below, but unbounded from above:

$$1 \leq \kappa(\mathbf{A}) < \infty \tag{10}$$

If the Euclidean norm is used in the above definition of $\kappa(\mathbf{A})$, which will be indicated in the form $\kappa_E(\mathbf{A})$, then it follows that

$$\kappa_E(\mathbf{A}^T\mathbf{A}) = \kappa_E^2(\mathbf{A}) \tag{11}$$

i.e., the roundoff error amplification incurred when solving a system of the form

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{b}^* \tag{12}$$

is the square of that incurred when solving system (1). In the above equation we have used a different right-hand side from that of eq.(1) to be consistent with the different matrix coefficient at hand.

Upon comparing eq.(12) with the normal equations (4), it is apparent that $\mathbf{b}^*$ of the former equals $\mathbf{A}^T\mathbf{b}$ of the latter. As a consequence, then, the roundoff error incurred when solving the normal equations is bigger that that incurred when solving the original eq.(1), by a factor of $\kappa(\mathbf{A}) > 1$. The question that remains is how to solve eq.(1) *directly*, i.e., without passing through the normal equations, This question is answered in the balance of this subsection.

In other words, the straightforward evaluation of $\mathbf{A}^I$ is prone to ill-conditioning. Not only this, the same formula is computationally costly, 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 to obtain the said product. This means that $\mathbf{A}^T\mathbf{A}$ requires $n^2 \times q$ products and $n^2(q-1)$ additions. The computation of the right-hand side of that equation, additionally, involves n scalar products of two q-dimensional vectors, which amounts to an additional $q \times n$ multiplications and $(q-1)n$ additions.

An alternative method is used here (Golub and Van Loan 1983), which applies *Householder reflections*, as described below. First, both sides of eq.(1) are premultiplied by n Householder reflections $\mathbf{H}_i$, for $i = 1, \ldots, n$, i.e.,

$$\mathbf{HAx} = \mathbf{Hb} \tag{13}$$

where $\mathbf{H} = \mathbf{H}_n, \mathbf{H}_{n-1}, \ldots, \mathbf{H}_1$, the set $\{\,\mathbf{H}_i\,\}_1^n$ being chosen so that

$$\mathbf{HA} = \begin{bmatrix} \mathbf{U} \\ \mathbf{O} \end{bmatrix}, \quad \mathbf{Hb} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} \tag{14}$$

in which $\mathbf{U}$ is an $n \times n$ upper-triangular matrix and $\mathbf{O}$ is the $(q-n) \times n$ zero matrix, while $\mathbf{b}_U$, and $\mathbf{b}_L$ are n-dimensional and $(q-n)$-dimensional vectors. Thus, eq.(13) leads to two subsystems of equations:

$$\mathbf{Ux} = \mathbf{b}_U \tag{15a}$$

$$\mathbf{Ox} = \mathbf{b}_L \neq \mathbf{0} \tag{15b}$$

The least-square solution can be readily calculated by backward substitution from eq.(15a), and symbolically expressed as

$$\mathbf{x}_L = \mathbf{U}^{-1}\mathbf{b}_U. \tag{16}$$

Furthermore, eq.(15b) expresses a contradiction: The left-hand side is the product of the $(q-n) \times n$ zero matrix times the unknown vector, while the right-hand side is not necessarily zero. This subsystem thus yields the least-square error associated with the solution $\mathbf{x}_L$, namely, $\|\mathbf{b}_L\|$. The algorithm reducing eq.(1) to the form of eqs.(15a & b) is available in the literature on numerical analysis (Golub and Van Loan, 1983), and hence, need not be further discussed here. Suffice it to say that eq.(13) and eqs.(15a & b) in consequence, are identical, if expressed in a coordinate frame that has undergone various reflections. The latter, like rigid-body rotations, preserve the magnitudes of the q-dimensional vector making up the columns of $\mathbf{A}$ as well as the angles between any pair of these n vectors. As a consequence, the two matrices $\mathbf{A}$ and $\mathbf{HA}$ have identical condition numbers, when these are defined in terms of either the Euclidean or the Frobenius norms.

3.1.1 The Case of a Rank-Deficient A Matrix

If $\mathbf{A}$ is rank-deficient, namely, if $\text{rank}(\mathbf{A}) = r < n$, then, upon premultiplying eq.(1) by r Householder matrices, $\mathbf{H}_r$, $\mathbf{H}_{r-1}$, ...,$\mathbf{H}_1$, one can reduce eq.(1) to the form

$$\begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \\ \mathbf{O}_1 & \mathbf{O}_2 \end{bmatrix} \begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_{n-r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} \tag{17}$$

where $\mathbf{U}_1$, is a full-rank $r \times r$ upper-triangular matrix, $\mathbf{U}_2$ an $r \times (n-r)$ matrix that is, in general, different from zero, $\mathbf{x}_r$ and $\mathbf{x}_{n-r}$ being the r- and $(n-r)$-dimensional arrays of the corresponding partitioning of $\mathbf{x}$. Moreover, $\mathbf{O}_1$ and $\mathbf{O}_2$ are the $(q-r)\times r$ and $(q-r)\times(n-r)$ zero matrices, $\mathbf{b}_1$ and $\mathbf{b}_2$ being r- and $(q-r)$-dimensional vectors, respectively, with $\mathbf{b}_2 \neq \mathbf{0}$. The first r equations of eq.(17), namely,

$$[\,\mathbf{U}_1 \quad \mathbf{U}_2\,] \begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_{n-r} \end{bmatrix} = \mathbf{b}_1, \tag{18}$$

form an *underdetermined* system, to be studied in Subsection 3.2, and hence, admit infinitely-many solutions with zero least-square error, while the remaining equations, i.e.,

$$[\mathbf{O}_1, \mathbf{O}_2] \begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_{n-r} \end{bmatrix} = \mathbf{b}_2 \tag{19}$$

cannot hold for any $\mathbf{x} \neq \mathbf{0}$, which implies that eq.(19) is *approximated* with the least-square error $\|\mathbf{b}_2\|$ for any $\mathbf{x}$ verifying eq.(18). Now, $\mathbf{x}$ is to be determined from eq.(18), but it should be noted that this equation represents an underdetermined system of r equations in $n > r$ unknowns. How to find a meaningful unique solution to this system is the subject of Subsection 3.2. On the other hand, if all equations in system (1) happen to be consistent, then $\mathbf{b}_2$ is zero, which implies that the given system is not really overdetermined but underdetermined or determined, with a few redundant, compatible equations.

3.1.2 Example: Minimization of the Design Error of a Planar Four-Bar Linkage

The aim of the example is to determine the value of *Freudenstein parameters* (Freudenstein, 1955) k_1, k_2, and k_3 of a four-bar function-generating linkage, as shown in Fig. 1, producing a prescribed set of q input-output (I/O) pairs, $\{\psi_i, \phi_i\}_1^q$, such that the Euclidean norm of the *design error* reaches its minimum. The relations between the Freudenstein parameters and the link lengths are

$$
\begin{aligned}
k_1 &= \frac{(a_1^2 + a_2^2 - a_3^2 + a_4^2)}{2a_2a_4}, \quad k_2 = \frac{a_1}{a_2}, \quad k_3 = \frac{a_1}{a_4} \\
a_2 &= \frac{a_1}{k_1}, \quad a_3 = \frac{\sqrt{k_2^2 + k_3^2 + k_2^2k_3^2 - 2k_1k_2k_3}}{|k_2k_3|}, \quad a_4 = \frac{a_1}{k_3}
\end{aligned}
\tag{20}
$$

for a given value of a_1.

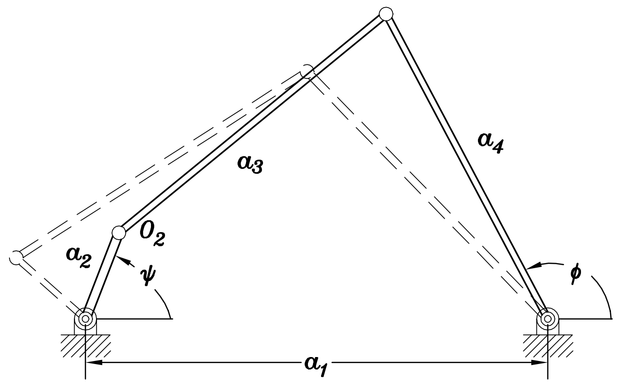

Figure 1: A four-bar linkage

Let ψ_i and ϕ_i represent the i^{th} input and output variables. The I/O equations for the planar four-bar linkage can be written in the form (Liu and Angeles, 1993):

$$\mathbf{Sk} = \mathbf{b} \tag{21}$$

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

$$
\mathbf{S} = \begin{bmatrix} 1 & \cos\phi_1 & -\cos\psi_1 \\ 1 & \cos\phi_2 & -\cos\psi_2 \\ \vdots & \vdots & \vdots \\ 1 & \cos\phi_q & -\cos\psi_q \end{bmatrix} \quad \mathbf{k} = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} \cos(\psi_1 - \phi_1) \\ \cos(\psi_2 - \phi_2) \\ \vdots \\ \cos(\psi_q - \phi_q) \end{bmatrix} \tag{22}
$$

The design error vector $\mathbf{d}$ is defined, in turn, as

$$\mathbf{d} = \mathbf{S}\mathbf{k} - \mathbf{b} \tag{23}$$

the objective function f thus being

$$f = \frac{1}{2}\mathbf{d}^T\mathbf{W}\mathbf{d} \tag{24}$$

where $\mathbf{W}$ is a positive-definite weighting matrix that is set here equal to $(1/q)\mathbf{1}$. When comparing the foregoing objective function f with the Euclidean norm of error $\mathbf{e}$ of eq.(3), we note that they differ by matrix $\mathbf{W}$. For this reason, minimizing function f is sometimes referred to as a *weighted least-square problem.* Note that we have normalized the objective function f so that it yields the *root-mean square* (rms) value of vector $\mathbf{d}$, and hence, the rms of the errors in the approximation of the synthesis equations (21). Hence, the problem can be treated as finding the least-square solution of eq.(1), with matrix $\mathbf{A}$ defined as

$$\mathbf{A} = \frac{1}{\sqrt{q}}\mathbf{S} \tag{25}$$

and vector $\mathbf{b}$ replaced by $(1/\sqrt{q})\mathbf{b}$. A set of input-output pairs $\{\psi_i, \phi_t\}_1^{10}$ that the linkage is required to generate is included in Table 1.

Table 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 synthesis matrix $\mathbf{S}$ and vector $\mathbf{b}$ are calculated as

$$
\mathbf{S} = \frac{1}{\sqrt{10}} \begin{bmatrix} 1.0000 & -0.0299 & 0.5573 \\ 1.0000 & -0.0348 & 0.6499 \\ 1.0000 & -0.0493 & 0.7337 \\ 1.0000 & -0.0735 & 0.8076 \\ 1.0000 & -0.1073 & 0.8706 \\ 1.0000 & -0.1506 & 0.9219 \\ 1.0000 & -0.2031 & 0.9606 \\ 1.0000 & -0.2643 & 0.9864 \\ 1.0000 & -0.3337 & 0.9988 \\ 1.0000 & -0.4102 & 0.9977 \end{bmatrix}, \quad \mathbf{b} = \frac{1}{\sqrt{10}} \begin{bmatrix} 0.8466 \\ 0.7822 \\ 0.7148 \\ 0.6474 \\ 0.5825 \\ 0.5219 \\ 0.4671 \\ 0.4193 \\ 0.3793 \\ 0.3477 \end{bmatrix} \tag{26}
$$

with the least-square solution given by

$$
\mathbf{k} = [1.2790 \quad 0.4727 \quad -0.7423]^T \tag{27}
$$

which yields $a_1 = 1.0$, $a_2 = 2.1155$, $a_3 = 3.8184$, and $a_4 = -1.3472$, with the corresponding optimum linkage shown in Fig. 2. In this figure the linkage is shown at a configuration whereby $\psi = \psi_1$ and $\phi = \phi_1 + \pi + \epsilon_1$, with ϵ_1 referring to the approximation error in ϕ, which is *different* from d_1, the first component of vector $\mathbf{d}$ of eq.(29). The reason why we added π to ϕ_1 is that $a_4 < 0$ (Denavit and Hartenberg, 1964).

The least-square design error is

$$
\begin{aligned}
\mathbf{d} = [\; & 0.0045 \quad -0.0020 \quad -0.0038 \quad -0.0027 \quad -0.0005 \\
& 0.0016 \quad 0.0028 \quad 0.0025 \quad 0.0005 \quad -0.0032 \;]^T
\end{aligned} \tag{28}
$$

And hence, its rms value is

$$
d_{rms} = 0.0027 \tag{29}
$$

3.2 The Minimum-Norm Solution of Underdetermined Linear Systems

An underdetermined system of linear equations is defined as

$$
\mathbf{C}\mathbf{x} = \mathbf{d} \tag{30}
$$

where $\mathbf{C}$ is a $l \times n$ matrix with $l < n$. Moreover, it is assumed that all equations of the linear system (30) are consistent. With this assumption, it is apparent that there

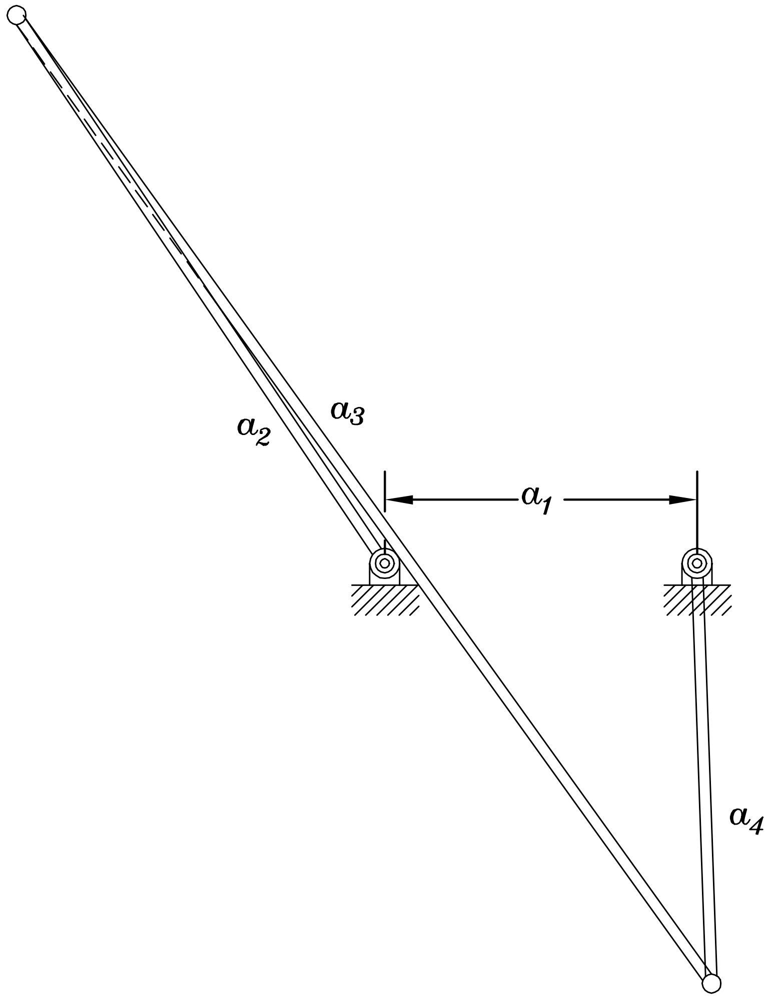

Figure 2: The optimum four-bar linkage at $\psi = \psi_1$

exist infinitely many solutions to eq.(30), among which a unique solution, denoted as $\mathbf{x}_0$, can be found that has a minimum Euclidean norm, and hence, is termed the *minimum-norm solution* of that equation. If matrix $\mathbf{C}$ is of full-rank, the said solution can be expressed as

$$\mathbf{x}_0 = \mathbf{C}^\dagger \mathbf{d} \tag{31a}$$

where

$$\mathbf{C}^\dagger = \mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1} \tag{31b}$$

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.(31a), 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.(5b). Moreover, the solution of eq.(31a) does not hold when $\mathbf{C}$ is rank-deficient. An efficient and robust alternative is to apply Householder reflections as follows (Golub and Van Loan 1983): First, a set of $n \times n$ Householder reflections $\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_l$ is defined, such that

$$\mathbf{C}\mathbf{H}^T = [\,\mathbf{L} \quad \mathbf{O}\,] \tag{32}$$

where $\mathbf{H} = \mathbf{H}_l, \mathbf{H}_{l-1}, \ldots, \mathbf{H}_1$. Moreover, $\mathbf{L}$ is a $l \times l$ lower-triangular matrix and $\mathbf{O}$ is the $l \times (n-l)$ zero matrix. Next, eq.(30) is rewritten in the form

$$\mathbf{C}\mathbf{H}^T\mathbf{H}\mathbf{x} = \mathbf{d} \tag{33}$$

which does not alter the original equation (30) because $\mathbf{H}$ is orthogonal. Letting $\mathbf{y} = \mathbf{H}\mathbf{x}$, from eqs. (32) and (33), 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, $\mathbf{y}$ will be the minimum-norm solution of the underdetermined system (30) if $\mathbf{y}$ is, correspondingly, the minimum-norm solution of the system

$$(\mathbf{H}\mathbf{C}^T)^T\mathbf{y} = \mathbf{d} \tag{34}$$

and hence, $\mathbf{y}$ takes on the form:

$$\mathbf{y} = \begin{bmatrix} \mathbf{L}^{-1}\mathbf{d} \\ \mathbf{0} \end{bmatrix} \equiv \mathbf{y}_0 \tag{35}$$

i.e., the last $(n-l)$ components of $\mathbf{y}_0$ are zero. In this way, $\mathbf{y}_0$ verifies eq.(33) and has a minimum-norm. Then, the minimum-norm solution $\mathbf{x}_0$ can be readily computed as

$$\mathbf{x}_0 = \mathbf{H}^T\mathbf{y}_0. \tag{36}$$

3.2.1 The Case of a Rank-Deficient C Matrix

If $\mathbf{C}$ is rank-deficient, namely, if $\text{rank}(\mathbf{C}) = r < l$, 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 eq.(34) becomes

$$\begin{bmatrix} \mathbf{L}_1 & \mathbf{O}_1 \\ \mathbf{L}_2 & \mathbf{O}_2 \end{bmatrix} \begin{bmatrix} \mathbf{y}_r \\ \mathbf{y}_{n-r} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix} \tag{37}$$

where $\mathbf{L}_1$ is a $r \times r$ lower-triangular matrix, $\mathbf{L}_2$ a $(l-r) \times r$ nonzero matrix that, in general, is not zero, $\mathbf{y}_r$ and $\mathbf{y}_{n-r}$ being the r- and $(n-r)$-dimensional arrays of the corresponding partitioning of $\mathbf{y}$. Moreover, $\mathbf{O}_1$ and $\mathbf{O}_2$ are the $r \times (n-r)$ and

$(l-r) \times (n-r)$ zero matrices. Furthermore, $\mathbf{d}$ has been partitioned into the r- and $(l-r)$-dimensional vectors $\mathbf{d}_1$ and $\mathbf{d}_2$, respectively. Based on the assumption that all equations of system (30) are consistent, the last $(l-r)$ equations of system (37) depend linearly on the first r equations and, hence, are redundant. Therefore, the minimum-norm solution of $\mathbf{y}$ takes the form

$$\mathbf{y}_0 = \begin{bmatrix} \mathbf{L}_1^{-1}\mathbf{d}_1 \\ \mathbf{0} \end{bmatrix} \tag{38}$$

in which $\mathbf{0}$ denotes the $(n-r)$-dimensional zero vector, while $\mathbf{x}_0$ is still obtained from eq.(36).

3.2.2 Example: 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

$$\mathbf{a} \times \mathbf{x} = \mathbf{b} \tag{39}$$

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:

$$a_2x_3 - a_3x_2 = b_1 \tag{40a}$$
$$a_3x_1 - a_1x_3 = b_2 \tag{40b}$$
$$a_1x_2 - a_2x_1 = b_3 \tag{40c}$$
$$\tag{40d}$$

Note that the foregoing equation can be cast in the form of eq.(1) if we define matrix $\mathbf{A}$ as

$$\mathbf{A} \equiv \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{41}$$

which is apparently skew-symmetric, i.e.,

$$\mathbf{A}^T = -\mathbf{A} \tag{42}$$

Picking up, for example, the first two scalar equations above, we obtain an underdetermined system of the form (30), with

$$\mathbf{C} \equiv \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{43}$$

and hence, the corresponding minimum-norm solution $\mathbf{x}_0$ is given by eqs.(31a & b), with

$$\mathbf{C}\mathbf{C}^T = \begin{bmatrix} a_2^2 + a_3^2 & -a_1 a_2 \\ -a_1 a_2 & a_1^2 + a_3^2 \end{bmatrix} \tag{44}$$

Hence,

$$(\mathbf{C}\mathbf{C}^T)^{-1} = \frac{1}{\Delta} \begin{bmatrix} a_1^2 + a_3^2 & a_1 a_2 \\ a_1 a_2 & a_2^2 + a_3^2 \end{bmatrix} \tag{45}$$

where

$$\Delta \equiv \det(\mathbf{C}\mathbf{C}^T) = (a_2^2 + a_3^2)(a_1^2 + a_3^2) - a_1^2 a_2^2 > 0 \tag{46}$$

a relation that the reader can readily prove. Therefore,

$$\mathbf{C}^\dagger = \frac{1}{\Delta} \begin{bmatrix} a_1 a_2 a_3 & (a_2^2 + a_3^2) a_3 \\ -(a_1^2 + a_3^2) a_3 & -a_1 a_2 a_3 \\ a_2 a_3^2 & -a_1 a_3^2 \end{bmatrix} \tag{47}$$

and

$$\mathbf{x}_0 = \frac{1}{\Delta} \begin{bmatrix} a_1 a_2 a_3 b_1 + (a_2^2 + a_3^2) a_3 b_2 \\ -(a_1^2 + a_3^2) a_3 b_1 - a_1 a_2 a_3 b_2 \\ a_2 a_3^2 b_1 - a_1 a_3^2 b_2 \end{bmatrix} \tag{48}$$

Alternatively, we can solve the problem in a more elegant, and robust, way. Note that the foregoing solution has an element of arbitrariness that may lead to ill-conditioning: There is no guarantee that the two equations chosen are the best choice from the condition-number view point. 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.,

$$\mathbf{a}^T\mathbf{x} = 0 \tag{49}$$

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

$$\mathbf{M}\mathbf{x} = \mathbf{n} \tag{50}$$

where $\mathbf{M}$ and $\mathbf{n}$ are given by

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} \\ \mathbf{a}^T \end{bmatrix}, \quad \mathbf{n} = \begin{bmatrix} \mathbf{b} \\ 0 \end{bmatrix} \tag{51}$$

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

$$\mathbf{x}_L = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{n} \tag{52}$$

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:

$$\mathbf{M}^T\mathbf{M} = [\,\mathbf{A}^T \quad \mathbf{a}\,] \begin{bmatrix} \mathbf{A} \\ \mathbf{a}^T \end{bmatrix} = \mathbf{A}^T\mathbf{A} + \mathbf{a}\mathbf{a}^T \tag{53}$$

But, since $\mathbf{A}$ is skew-symmetric,

$$\mathbf{M}^T\mathbf{M} = -\mathbf{A}^2 + \mathbf{a}\mathbf{a}^T \tag{54}$$

As the reader can readily verify,

$$\mathbf{A}^2 = -\|\mathbf{a}\|^2\mathbf{1} + \mathbf{a}\mathbf{a}^T \tag{55}$$

Hence,

$$\mathbf{M}^T\mathbf{M} = \|\mathbf{a}\|^2\mathbf{1} \tag{56}$$

Therefore,

$$(\mathbf{M}^T\mathbf{M})^{-1} = \frac{1}{\|\mathbf{a}\|^2}\mathbf{1} \tag{57}$$

That is ,

$$\mathbf{x}_L = \frac{1}{\|\mathbf{a}\|^2}\mathbf{1}\,[\,\mathbf{A}^T \quad \mathbf{a}\,] \begin{bmatrix} \mathbf{b} \\ 0 \end{bmatrix} = \frac{1}{\|\mathbf{a}\|^2}\mathbf{A}^T\mathbf{b} \tag{58}$$

which can be further expressed as

$$\mathbf{x}_L = -\frac{\mathbf{a} \times \mathbf{b}}{\|\mathbf{a}\|^2} \tag{59}$$

thereby obtaining a much simpler expression than that displayed above as $\mathbf{x}_0$.

4 Unconstrained Nonlinear Least-Square Problems

The problem at hand is defined as: *Find an approximate solution, with the least-square error, to the overdetermined system of nonlinear equations shown below*:

$$\boldsymbol{\phi}(\mathbf{x}) = \mathbf{0} \tag{60}$$

where $\boldsymbol{\phi}$ and $\mathbf{x}$ are q- and n-dimensional vectors, respectively, with $q > n$. The problem can be described as

$$f(\mathbf{x}) = \frac{1}{2}\boldsymbol{\phi}^T\mathbf{W}\boldsymbol{\phi} \quad \rightarrow \quad \min_{\mathbf{x}} \tag{61}$$

with $\mathbf{W}$ defined as a $q \times q$ positive-definite weighting matrix. The solution is computed iteratively, using the Newton-Gauss method, as described below: First, an initial guess of $\mathbf{x}$ is given, say $\mathbf{x}^0$; then, a sequence, $\mathbf{x}^1$, $\mathbf{x}^2$, ..., is generated, where

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k \tag{62}$$

such that $\Delta\mathbf{x}^k$ is the least-square solution of the unconstrained overdetermined linear system

$$\mathbf{V}\boldsymbol{\Phi}(\mathbf{x}^k)\Delta\mathbf{x}^k = -\mathbf{V}\boldsymbol{\phi}(\mathbf{x}^k) \tag{63}$$

with $\boldsymbol{\Phi}$ defined as the $q \times n$ gradient matrix of the vector function $\boldsymbol{\phi}$, i.e.,

$$\boldsymbol{\Phi}(\mathbf{x}) = \frac{\partial\boldsymbol{\phi}(\mathbf{x})}{\partial\mathbf{x}} \tag{64}$$

and $\mathbf{V}$ is the $q \times q$ lower-triangular Cholesky factor of $\mathbf{W}$ in the sense of $\mathbf{W} = \mathbf{V}^T\mathbf{V}$. Equation (63) can thus be solved for a $\Delta\mathbf{x}^k$ that verifies these equations with the least-square error, namely,

$$\Delta\mathbf{x} = -(\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\phi} \tag{65}$$

where the superscripts have been dropped for simplicity. What we need now is a criterion to stop the procedure. To obtain this criterion, we derive the condition under which $f(\mathbf{x})$, as defined in eq.(61), attains a stationary value, which occurs when its gradient $\partial f/\partial\mathbf{x}$ vanishes. The above gradient is readily calculated by application of the chain rule, namely,

$$\frac{\partial f}{\partial\mathbf{x}} = \left(\frac{\partial\boldsymbol{\phi}}{\partial\mathbf{x}}\right)^T\frac{\partial f}{\partial\boldsymbol{\phi}} \tag{66}$$

where

$$\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x}} = \boldsymbol{\Phi}, \quad \frac{\partial f}{\partial \boldsymbol{\phi}} = \mathbf{W}\boldsymbol{\phi} \tag{67}$$

and hence, the condition for a stationary point is

$$\boldsymbol{\Phi}^T \mathbf{W}\boldsymbol{\phi} = \mathbf{0} \tag{68}$$

which is the *normality condition* of the problem at hand. What this condition states is that, at a stationary point of f, $\boldsymbol{\phi}(\mathbf{x})$ need not vanish, but *must* lie in the nullspace of $\boldsymbol{\Phi}^T\mathbf{W}$. Now, from eq.(65) it is apparent that, at a stationary point, $\Delta\mathbf{x}$ vanishes, the stopping criterion sought thus being

$$\|\Delta\mathbf{x}\| < \epsilon \tag{69}$$

where ϵ is a prescribed tolerance. However, the normality condition alone does not guarantee a minimum, but only a stationary point. It turns out that, if the procedure converges, then it does so to a minimum, and neither to a maximum nor to a saddle point. Indeed, the sequence $f(\mathbf{x}^0)$, $f(\mathbf{x}^1)$, $\ldots$, $f(\mathbf{x}^k)$, $f(\mathbf{x}^{k+l})$, $\ldots$, obtained from the aforementioned sequence of $\mathbf{x}$ values evolves, to a first order, as $\Delta f(\mathbf{x})$:

$$\Delta f = \left(\frac{\partial f}{\partial \mathbf{x}}\right)^T \Delta\mathbf{x} \tag{70}$$

i.e.,

$$\Delta f = \boldsymbol{\phi}^T \mathbf{W}\boldsymbol{\Phi}\Delta\mathbf{x} \tag{71}$$

Upon substitution of $\Delta\mathbf{x}$, as given by eq.(65), into eq.(71), we obtain

$$\Delta f = -\boldsymbol{\phi}^T \mathbf{W}\boldsymbol{\Phi}(\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\phi} \tag{72}$$

which is, apparently, negative-definite, and hence, to a first-order approximation, the sequence of f values decreases monotonically. In the neighbourhood of a stationary point the first-order approximation is good enough, and hence, if the procedure converges, it does so to a minimum.

Notice that if $q = n$, then we have a determined system of nonlinear equations (60). This system can most efficiently be solved, numerically, using the Newton-Raphson method (Kahaner, Moler, and Nash, 1989). However, the foregoing procedure, i.e., the Newton-Gauss method, can also be used to solve such a problem as illustrated in the first example below.

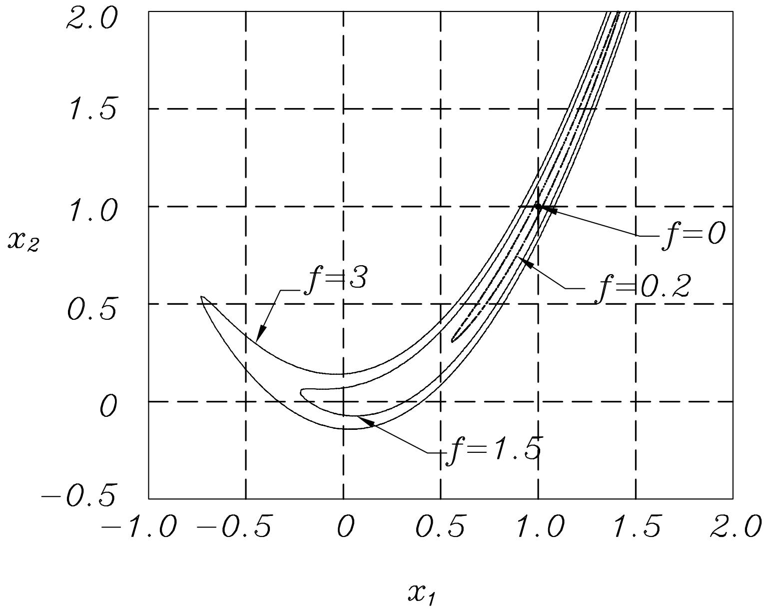

Figure 3: The contours of the Rosenbrock (a.k.a. the banana) function

4.1 Example 1: 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

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{73}$$

The problem can be treated as finding an approximate solution of the following system of nonlinear equations

$$\boldsymbol{\phi} = \begin{bmatrix} x_2 - x_1^2 \\ 1 - x_1 \end{bmatrix} \tag{74}$$

such that the least-square error f is a minimum, i.e.,

$$f(\mathbf{x}) = \frac{1}{2}\boldsymbol{\phi}^T \mathbf{W} \boldsymbol{\phi} \quad \rightarrow \quad \min_{x_1,\, x_2} \tag{75}$$

with

$$\mathbf{W} = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix} \tag{76}$$

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 2. 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. 3. Now we verify the normality condition (68). Since $\boldsymbol{\phi} = \mathbf{0}$ at x_{opt}, this condition is readily verified.

Table 2: Solutions of the Rosenbrock function

i	1	2	3
$\mathbf{x}_i$	$[1.000000,\ 0.360000]^T$	$[1.000000,\ 1.000000]^T$	$[1.000000,\ 1.000000]^T$

4.2 Example 2: Finding the "Intersection" of a Circle, a Hyperbola, and a Parabola

While two planar curves can intersect in a finite set of points, three such curves are unlikely to intersect at all. However, the point(s) closest to the three (or more) curves can be found using the concepts developed in this section. We include here an example to illustrate this idea. The problem at hand consists in finding the points closest to all three curves, a circle, a hyperbola and a parabola, shown in Fig. 4, and given as

$$\phi_1(x, y) = x^2 + y^2 - 4 = 0 \tag{77a}$$

$$\phi_2(x, y) = x^2 - y^2 - 1 = 0 \tag{77b}$$

$$\phi_3(x, y) = x^2 - 2.4y = 0 \tag{77c}$$

$$\tag{77d}$$

The problem is formulated as an unconstrained nonlinear least-square problem. Solutions are found and listed in Table 3 by using two different initial guesses.

Table 3: Solutions of a circle intersecting a hyperbola

Initial Guess	[0.6, 0.6]	[−0.9, 0.6]
Solution	[1.615664, 1.178818]	[−1.615664, 1.178818]
N	5	5
f	0.048302	0.048302

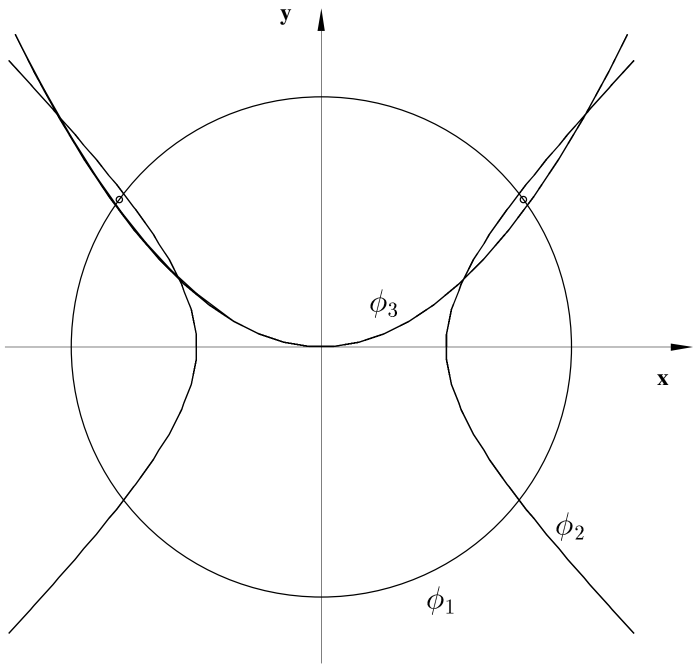

Figure 4: A circle, a hyperbola, and a parabola

5 Equality-Constrained Linear Least-Square Problems — The Orthogonal-Decomposition Algorithm

Given the system of linear equations

$$\mathbf{Ax} = \mathbf{b} \tag{78}$$

find a vector $\mathbf{x}$ that verifies the above system with the least-square error, which is defined as

$$f \equiv \frac{1}{2}(\mathbf{Ax} - \mathbf{b})^T \mathbf{W}(\mathbf{Ax} - \mathbf{b}) \rightarrow \min_{\mathbf{x}} \tag{79}$$

subject to the linear constraints

$$\mathbf{Cx} = \mathbf{d} \tag{80}$$

Here, $\mathbf{x}$ is the n-dimensional vector of design variables, while $\mathbf{A}$ and $\mathbf{C}$ are $q \times n$ and $l \times n$ matrices, while $\mathbf{b}$ and $\mathbf{d}$ are q- and l-dimensional vectors. Moreover, $\mathbf{W}$

is a $q \times q$ positive-definite weighting matrix, with q, l and n subject to

$$q + l > n \quad \text{and} \quad l < n \tag{81}$$

Note that the first of the foregoing inequalities excludes the possibility of a unique solution upon solving for $\mathbf{x}$ from eqs.(78) and (80); the second inequality excludes the possibility of overconstraining the problem.

If $\mathbf{A}$ and $\mathbf{C}$ are full-rank matrices, then the forgoing problem has a *unique* solution, namely,

$$\mathbf{x} = \mathbf{PQb} + \mathbf{Rd} \tag{82}$$

where $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{R}$ are the $n \times n$-, $n \times q$- and $n \times l$ matrices that follow:

$$\mathbf{P} = \mathbf{1}_n - \mathbf{RC} \tag{83}$$

$$\mathbf{Q} = (\mathbf{A}^T\mathbf{WA})^{-1}\mathbf{A}^T\mathbf{W} \tag{84}$$

$$\mathbf{R} = (\mathbf{A}^T\mathbf{WA})^{-1}\mathbf{C}^T[\mathbf{C}(\mathbf{A}^T\mathbf{WA})^{-1}\mathbf{C}^T]^{-1} \tag{85}$$

and $\mathbf{1}_n$ is the $n \times n$ identity matrix.The above solution is nevertheless unsuitable for numerical implementation. It contains inversions of products of several matrices times their transposes, which produces frequently ill-conditioning (Kahaner, Moler, and Nash, 1989). An alternative approach to obtain the solution under study could be to partition $\mathbf{C}$ into a $l \times l$ and a $l \times (n-l)$ submatrices, where care should be taken so as to choose a well-conditioned $l \times l$ matrix, for safe inversion. Correspondingly, vector $\mathbf{x}$ should be partitioned into a *master* part $\mathbf{x}_M$, of $n - l$ components, and a *slave* part $\mathbf{x}_S$ of l 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 $n - l$. However, an arbitrary partitioning of $\mathbf{C}$ may lead to an ill-conditioned $l \times l$ 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

$$N = \frac{n!}{l!(n-l)!} \tag{86}$$

and hence, N can become quite large, even for modest values of n and l. 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 l equations in l 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 (82). Indeed, vector $\mathbf{Qb}$ of that solution represents the unconstrained least-square approximation of eq.(79). The second term of the right-hand side of eq.(82) is the minimum-norm solution of the underdetermined system (80), based on the norm defined as

$$\|\mathbf{x}\|_W^2 = \mathbf{x}^T\mathbf{A}^T\mathbf{W}\mathbf{A}\mathbf{x} \tag{87}$$

Thus, $\mathbf{P}$ is a *projector* 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. Thus the range and the nullspace of $\mathbf{C}$ appear to play an important role in the solution of the foregoing problem. Moreover, the range and the nullspace of any matrix representing a linear transformation of $\mathbb{R}^n$ are orthogonal subspaces 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 $\mathbf{C}^T$ and a second one lying in the nullspace of $\mathbf{C}$. Now let $\mathbf{L}$ be an $n \times (n-l)$ matrix spanning the nullspace of $\mathbf{C}$, i.e.,

$$\mathbf{CL} = \mathbf{O}_{ln'} \tag{88}$$

where $\mathbf{O}_{ln'}$ represents the $l \times (n-l)$ 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,

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{x}_U \tag{89}$$

in which $\mathbf{x}_0$ represents the minimum-norm solution to the constraint equation (80), 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 evaluated by means of the method discussed in Subsection 3.2.

We define a $q \times q$ matrix $\mathbf{V}$ to be the Cholesky factor of the given weighting matrix $\mathbf{W}$, i.e.,

$$\mathbf{W} = \mathbf{V}^T\mathbf{V}.$$

With $\mathbf{x}_0$ known, $\mathbf{x}_U$ is found as the least-square approximation of

$$\mathbf{VAx}_U = \mathbf{V}(\mathbf{b} - \mathbf{Ax}_0) \tag{90}$$

subject to the constraints

$$\mathbf{Cx}_U = \mathbf{0} \tag{91}$$

Moreover, we define

$$\mathbf{x}_U = \mathbf{Lu} \tag{92}$$

where $\mathbf{L}$ was introduced in eq.(88), and $\mathbf{u}$ is an $(n-l)$-dimensional vector. Equation (90) thus becomes

$$\mathbf{VALu} = \mathbf{V}(\mathbf{b} - \mathbf{Ax}_0) \tag{93}$$

Hence, it is apparent that $\mathbf{u}$ can be considered as the unconstrained least-square solution of the overdetermined system (93).

However, matrix $\mathbf{L}$, an orthogonal complement of $\mathbf{C}$, is not unique. We define a distinct $\mathbf{L}$ such that

$$\mathbf{H}^T\mathbf{L} = \begin{bmatrix} \mathbf{O} \\ \mathbf{1} \end{bmatrix} \tag{94}$$

where $\mathbf{1}$ is the $(n-l) \times (n-l)$ identity matrix and $\mathbf{O}$ is the $l \times (n-l)$ zero matrix. From equation (94), one can obtain matrix $\mathbf{L}$ without any additional computations, for

$$\mathbf{L} = \mathbf{H} \begin{bmatrix} \mathbf{O} \\ \mathbf{1} \end{bmatrix} \tag{95}$$

whence it is apparent that $\mathbf{L}$ is isotropic, i.e., its condition number is equal to unity. This means that the generalized inverse $\mathbf{L}^I$ can be computed without roundoff-error amplification. In fact, this inverse reduces to $\mathbf{L}^T$, for

$$\mathbf{L}^I = ([\mathbf{O}^T \quad \mathbf{1}]\mathbf{H}^T\mathbf{H} \begin{bmatrix} \mathbf{O} \\ \mathbf{1} \end{bmatrix})^{-1}[\mathbf{O}^T \quad \mathbf{1}]\mathbf{H}^T = \mathbf{L}^T \tag{96}$$

Once $\mathbf{L}$ is known, equation (93) can be solved for $\mathbf{u}$ as the least-square approximation of that system. Then, $\mathbf{x}_U$ is calculated from equation (92).

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 Equality-Constrained Nonlinear Least-Square Problems

The solution of nonlinear least-square problems by means of the ODA is now described: The problem consists of finding the least-square error f of an overdetermined system of nonlinear equations, $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{0}$, i.e.,

$$f(\mathbf{x}) = \frac{1}{2}\boldsymbol{\phi}^T\mathbf{W}\boldsymbol{\phi} \quad \rightarrow \quad \min_{\mathbf{x}} \tag{97}$$

subject to the nonlinear constraints

$$\mathbf{h}(\mathbf{x}) = \mathbf{0} \tag{98}$$

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.

With the introduction of a l-dimensional vector of Lagrange multipliers $\boldsymbol{\lambda}$, we turn the constrained problem into an unconstrained one. Indeed, by adjoining the constraint vector $\mathbf{h}$ to the objective function f, we obtain a new objective function F:

$$F \equiv f + \boldsymbol{\lambda}^T\mathbf{h} \quad \rightarrow \quad \min_{\mathbf{x},\boldsymbol{\lambda}} \tag{99a}$$

where the vector of design variables has been extended, for it is now of dimension $n + l$, and denoted by $\boldsymbol{\xi}$, i.e.,

$$\boldsymbol{\xi} \equiv \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix} \tag{99b}$$

The normality condition of the unconstrained problem (99a) is now

$$\frac{\partial F}{\partial \boldsymbol{\xi}} = \mathbf{0}_{n+l} \tag{100a}$$

with $\mathbf{0}_{n+l}$ denoting the $(n + l)$-dimensional zero vector. In terms of $\mathbf{x}$ and $\boldsymbol{\lambda}$, the above normality condition becomes

$$\frac{\partial f}{\partial \mathbf{x}} = \mathbf{0}_n, \quad \frac{\partial f}{\partial \boldsymbol{\lambda}} = \mathbf{0}_l \tag{100b}$$

in which $\mathbf{0}_n$ and $\mathbf{0}_l$ denote the n- and l-dimensional zero vectors. Moreover, the first of the above conditions becomes

$$\boldsymbol{\Phi}^T\mathbf{W}\boldsymbol{\phi} + \mathbf{J}^T\boldsymbol{\lambda} = \mathbf{0}_n \tag{101}$$

where $\mathbf{\Phi} = \mathbf{\Phi}(\mathbf{x})$ and $\mathbf{J} = \mathbf{J}(\mathbf{x})$ denote the $q \times n$ and the $l \times n$ gradient matrices of the vector functions $\boldsymbol{\phi}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$, namely,

$$\mathbf{\Phi}(\mathbf{x}) = \frac{\partial \boldsymbol{\phi}(\mathbf{x})}{\partial \mathbf{x}}, \quad \mathbf{J}(\mathbf{x}) = \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}} \tag{102}$$

The second normality condition of eq.(100b), in turn, is nothing but a restatement of the constraint, eq.(98). The solution of the problem at hand can be obtained iteratively: From an initial guess $\mathbf{x}^0$, *not necessarily feasible*, i.e., not necessarily verifying the constraints, and so, $\mathbf{h}(\mathbf{x}^0) \neq \mathbf{0}$, the sequence $\mathbf{x}^1$, $\mathbf{x}^2$, ..., $\mathbf{x}^k$, $\mathbf{x}^{k+1}$ is generated as

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k \tag{103}$$

The increment $\Delta\mathbf{x}^k$ is computed as the solution of a constrained least-square problem with a quadratic performance index and linear constraints, namely,

$$\min_{\Delta\mathbf{x}^k} \frac{1}{2} \mathbf{e}^{k^T} \mathbf{W} \mathbf{e}^k \tag{104}$$

subject to

$$\mathbf{J}(\mathbf{x}^k)\Delta\mathbf{x}^k = -\mathbf{h}(\mathbf{x}^k) \tag{105}$$

where $\mathbf{e}^k$ is defined as

$$\mathbf{e}^k \equiv -\boldsymbol{\phi}(\mathbf{x}^k) - \mathbf{\Phi}(\mathbf{x}^k)\Delta\mathbf{x}^k \tag{106}$$

Moreover, $\mathbf{\Phi}^k$ and $\mathbf{J}^k$ will be assumed to be of full rank throughout, the solution $\Delta\mathbf{x}^k$ thus being expressed as

$$\Delta\mathbf{x}^k = \mathbf{v}^k + \mathbf{L}^k\mathbf{u}^k \tag{107}$$

where $\mathbf{v}^k$ and $\mathbf{u}^k$ are the minimum-norm and the least-square solutions to an underdetermined and an overdetermined system, namely,

$$\mathbf{J}^k\mathbf{v}^k = -\mathbf{h}^k \tag{108}$$

$$\mathbf{B}^k\mathbf{L}^k\mathbf{u}^k = -\mathbf{V}^k\boldsymbol{\phi}^k - \mathbf{B}^k\mathbf{v}^k, \quad \mathbf{B}^k \equiv \mathbf{V}\mathbf{\Phi}^k \tag{109}$$

and $\mathbf{L}^k$ is an orthogonal complement of $\mathbf{J}^k$, to be defined presently.

The stopping criteria of the procedure are

$$||\Delta\mathbf{x}^k|| \leq \epsilon_1 \quad \text{and} \quad ||\mathbf{h}(\mathbf{x}^k)|| \leq \epsilon_2$$

for prescribed tolerances ϵ_1 and ϵ_2. These criteria are verified when the normality condition (101) *and the constraint* (98) are verified within the given tolerances. Moreover, from eqs. (108) and (109), $\Delta\mathbf{x}$ can be written as

$$\Delta\mathbf{x} = (\mathbf{M}\mathbf{\Phi}^T\mathbf{W}\mathbf{\Phi} - \mathbf{1}_n)\mathbf{J}^\dagger\mathbf{h} - \mathbf{M}\mathbf{\Phi}^T\mathbf{W}\boldsymbol{\phi} \tag{110}$$

where superscripts have been dropped 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

$$\mathbf{M} = \mathbf{L}(\mathbf{L}^T\mathbf{\Phi}^T\mathbf{W}\mathbf{\Phi}\mathbf{L})^{-1}\mathbf{L}^T \tag{111}$$

Upon convergence, the constraint equations (98) are verified, and hence, the first term of $\Delta\mathbf{x}$, as given by eq.(110), vanishes. Moreover, the normality condition (99a) holds, $\Delta\mathbf{x}$ taking on the form shown below, upon convergence:

$$\Delta\mathbf{x} = \mathbf{L}(\mathbf{L}^T\mathbf{\Phi}^T\mathbf{W}\mathbf{\Phi}\mathbf{L})^{-1}\mathbf{L}^T\mathbf{J}^T\boldsymbol{\lambda} \tag{112}$$

However, $\mathbf{L}$ is the $n \times (n-l)$ orthogonal complement of $\mathbf{J}$, namely,

$$\mathbf{J}\mathbf{L} = \mathbf{O}_{ln'} \tag{113}$$

with $\mathbf{O}_{ln'}$ defined already in eq.(88). Therefore, upon convergence, $\Delta\mathbf{x} \to \mathbf{0}$.

The sequence $\{\Delta\mathbf{x}^k\}$ produces a sequence $\{f^k\}$, the increment Δf between two consecutive values of the sequence being given by

$$\Delta f = (\nabla f)^T\Delta\mathbf{x} \tag{114}$$

where ∇f is the gradient of f, i.e., $\nabla f = \mathbf{\Phi}\mathbf{W}\boldsymbol{\phi}$, and hence,

$$\begin{aligned}\Delta f &= (\mathbf{\Phi}^T\mathbf{W}\boldsymbol{\phi})^T\Delta\mathbf{x} \\ &= -\boldsymbol{\phi}^T\mathbf{W}\mathbf{\Phi}\mathbf{M}\mathbf{\Phi}^T\mathbf{W}\boldsymbol{\phi} - \boldsymbol{\phi}^T\mathbf{W}\mathbf{\Phi}(\mathbf{1} - \mathbf{M}\mathbf{\Phi}^T\mathbf{W}\mathbf{\Phi})\mathbf{J}^\dagger\mathbf{h}\end{aligned} \tag{115}$$

From eq.(115), 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

$$\Delta(\mathbf{h}^T\mathbf{h}) = -2\mathbf{h}^T\mathbf{h} \tag{116}$$

which is negative definite. Therefore, the procedure gives a sequence of $\mathbf{x}$ values that approaches the constraints.

6.1 Example 1: Finding the Eigenvalues and Eigenvectors of a Symmetric Matrix

The problem of finding the eigenvalues and corresponding eigenvectors of a symmetric $n \times n$ matrix $\mathbf{M}$ is solved as a constrained nonlinear least-square problem: For $i = 1, 2, \cdots, n$, and $k = 1, 2, \cdots, i$, find λ_i and $\mathbf{x}_i$ such that

$$\lambda_i = \min_{\mathbf{x}_i} \frac{1}{2} \mathbf{x}_i^T \mathbf{M} \mathbf{x}_i \tag{117}$$

subject to

$$\mathbf{x}_k^T \mathbf{x}_i = \begin{cases} 0, & \text{if } k = 1,\ 2,\ \cdots,\ i-1; \\ 1, & \text{if } k = i. \end{cases} \tag{118}$$

where λ_i is the ith eigenvalue of matrix $\mathbf{M}$ and $\mathbf{x}_i$ is the corresponding eigenvector. In order to use the ODA package to solve the problem, we define, for $i = 1, 2, \cdots, n$:

$$q = n \qquad \text{and} \qquad l = i, \tag{119}$$

$$\mathbf{x} = \mathbf{x}_i, \tag{120}$$

$$\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}, \tag{121}$$

$$\mathbf{h}(\mathbf{x}) = [\, \mathbf{x}_1^T \mathbf{x} \quad \cdots \quad \mathbf{x}_{i-1}^T \mathbf{x} \quad \mathbf{x}^T \mathbf{x} - 1 \,]^T, \tag{122}$$

$$\mathbf{W} = \mathbf{M} \tag{123}$$

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 $i = 1, 2, \cdots, n$, ODA's subroutine LSSCNL 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

$$\mathbf{M} = \begin{bmatrix} 4 & 2 & 1 & 1 & 1 \\ 2 & 4 & 2 & 1 & 1 \\ 1 & 2 & 4 & 2 & 1 \\ 1 & 1 & 2 & 4 & 2 \\ 1 & 1 & 1 & 2 & 4 \end{bmatrix}. \tag{124}$$

We use the initial guess

$$\mathbf{x}^0 = [\, 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \quad 0.1 \,]^T. \tag{125}$$

The eigenvalues and the eigenvectors computed with ODA are listed in Table 4. In that table, the number of iterations that ODA took till convergence was reached with $\epsilon_1 = 0.0001$ and $\epsilon_2 = 0.0001$, is indicated.

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 4: Eigenvalues and eigenvectors of $\mathbf{M}$

6.2 Example 2: A quadratic Objective Function with a Quadratic Constraint

Consider an optimization problem with an objective function defined as

$$f(\mathbf{x}) = \frac{1}{2}(9x_1^2 - 8x_1x_2 + 3x_2^2) \quad \rightarrow \quad \min_{x_1, x_2} \tag{126}$$

subject to the condition

$$h(\mathbf{x}) = x_1^2 + x_2^2 - 1 = 0 \tag{127}$$

The objective function $f(\mathbf{x})$ can be factored as

$$f(\mathbf{x}) = \frac{1}{2}\boldsymbol{\phi}^T \mathbf{W} \boldsymbol{\phi} \tag{128}$$

with

$$\mathbf{W} = \begin{bmatrix} 9 & -4 \\ -4 & 3 \end{bmatrix} \quad \text{and} \quad \boldsymbol{\phi} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{129}$$

Taking $\mathbf{x}_0 = [2 \quad 2]^T$ as initial guess, we obtain the optimum $f_{min}(\mathbf{x}) = 0.5$ with $\mathbf{x}_{opt} = [0.447215 \quad 0.894426]^T$ in five iterations. The contours for different values of constant f and the constraint $h = 0$ are displayed in Fig. 5. The iterations till convergence are shown in Table 5.

Table 5: Solutions of the quadratic function with a quadratic constraint

i	$\mathbf{x}_i$
1	$[0.787500,\ 1.462500]^T$
2	$[0.509018,\ 1.011063]^T$
3	$[0.450881,\ 0.901197]^T$
4	$[0.447247,\ 0.894443]^T$
5	$[0.447215,\ 0.894426]^T$

7 Equality-Constrained Optimization with Arbitrary Objective Function

The problem to be solved is defined as:

$$f = f(\mathbf{x}) \quad \rightarrow \quad \min_{\mathbf{x}} \tag{130}$$

subject to the nonlinear equality constraints

$$\mathbf{h}(\mathbf{x}) = \mathbf{0} \tag{131}$$

where $\mathbf{x}$ is an n-dimensional vector of design variables, the objective function $f(\mathbf{x})$ being a nonlinear function of $\mathbf{x}$, not necessarily quadratic in the sense of Section 6, 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.

7.1 Gerschgorin Stabilization

The *Gerschgorin Theorem* 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

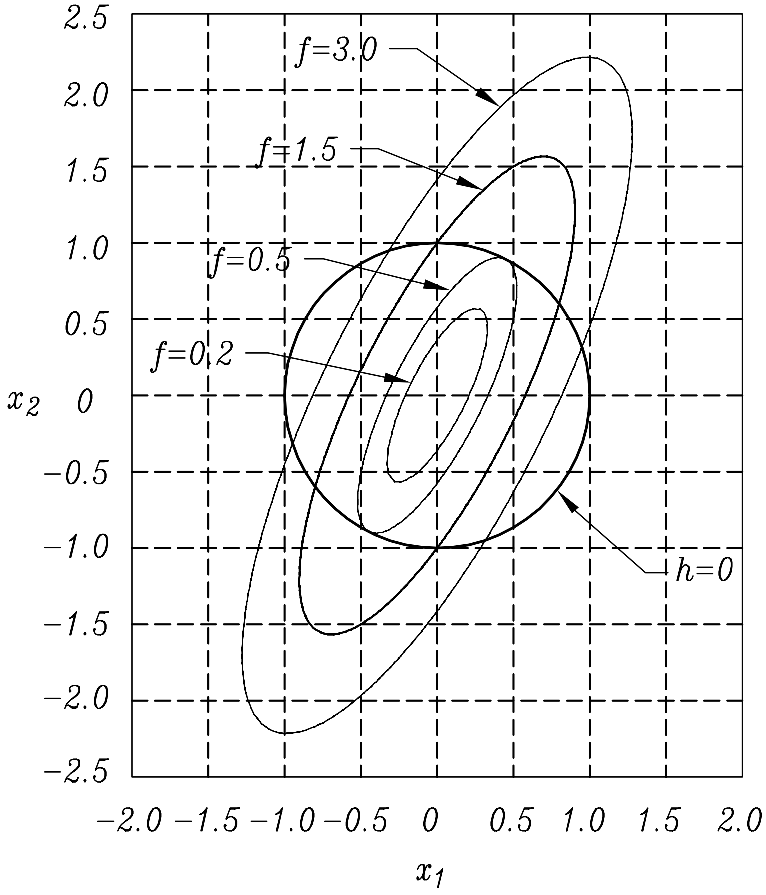

Figure 5: The contours of constant f and the constraint $h = 0$

$$r_i = \sum_{j=1, j\neq i}^{n} \mid a_{ij} \mid$$

in which $\mid \cdot \mid$ denotes the module of $(\cdot)$. The Gerschgorin Theorem is illustrated in Figure 6, 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

$$I_i = [a_{ii} - r_i, a_{ii} + r_i], \quad i = 1, 2, ..., n$$

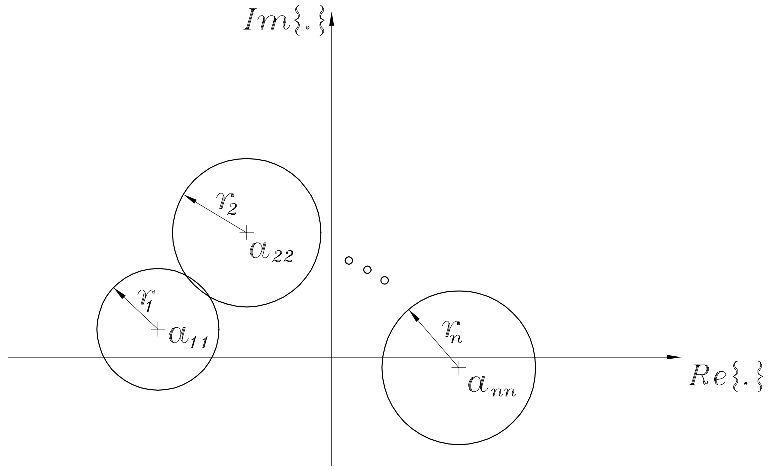

Figure 6: The Gerschgorin disks.

A lower bound l of the set $\{\lambda_i\}_1^n$ of eigenvalues of $\mathbf{A}$ is, thus,

$$l \equiv \min_i \{a_{ii} - r_i\}_1^n \tag{132}$$

the corresponding upper bound being

$$u \equiv \max_i \{a_{ii} + r_i\}_1^n \tag{133}$$

If $\mathbf{A}$ is positive-definite, all eigenvalues of $\mathbf{A}$ must be positive, which means that the lower bound l is 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.

Thus, in cases when the Hessian matrix $\nabla\nabla f$ of the objective function is not positive-definite, a *Gerschgorin-stabilization procedure* (Teng and Angeles, 1999) is applied by introducing a scalar $\mu > -l$, such that a new positive-definite matrix $\mathbf{H}$ is used to replace $\nabla\nabla f$, at the current iteration, with $\mathbf{H}$ defined as

$$\mathbf{H} = \nabla\nabla f + \mu\mathbf{1} \tag{134}$$

$\mathbf{H}$ is thus guaranteed to be positive-definite, for its minimum eigenvalue is *a fortiori* greater than zero. As in eq.(134), we will dispense henceforth with iteration subscripts whenever these are not essential.

7.2 Sequential Quadratic Programming with Gerschgorin Stabilization

With the problem defined in eq.(130), if the constraints in eq.(131) 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, under $\mathbf{x} = \mathbf{x}(\mathbf{u})$,

$$\mathbf{h}(\mathbf{x}(\mathbf{u})) = \mathbf{0} \tag{135}$$

If $\mathbf{h}(\mathbf{x}(\mathbf{u}))$ 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

$$f(\mathbf{x}^k + \Delta\mathbf{x}^k) \approx f(\mathbf{x}^k) + (\nabla f)_k^T \Delta\mathbf{x}^k + \frac{1}{2}(\Delta\mathbf{x}^k)^T(\nabla\nabla f)_k \Delta\mathbf{x}^k \to \min_{\Delta\mathbf{x}_k} \tag{136}$$

subject to

$$\mathbf{J}_k \Delta\mathbf{x}^k = -\mathbf{h}^k \tag{137}$$

Let

$$\Delta\mathbf{x}^k = \Delta\mathbf{x}_0^k + \mathbf{L}_k \Delta\mathbf{u}^k$$

where

$$\Delta\mathbf{x}_0^k = -\mathbf{J}_k^T(\mathbf{J}_k\mathbf{J}_k^T)^{-1}\mathbf{h}^k \tag{138}$$

is the minimum-norm solution of eq.(137), and $\mathbf{L}_k$ is the isotropic orthogonal complement of $\mathbf{J}_k$ defined in eqs.(94) and (95), while $\mathbf{J}_k$ itself is defined as the gradient of $\mathbf{h}$ with respect to $\mathbf{x}$, evaluated at $\mathbf{x} = \mathbf{x}_k$. $\mathbf{L}_k$ and $\Delta\mathbf{u}^k$ are thus found with the procedure described in Section 5 for solving constrained linear least-square problems. Moreover, with $\Delta\mathbf{x}_0^k$ given by eq.(138), $f(\mathbf{x}^k + \Delta\mathbf{x}^k)$ becomes a function solely of $\Delta\mathbf{u}^k$, i.e.,

$$\begin{aligned} f(\Delta\mathbf{u}^k) \approx \tilde{f}(\Delta\mathbf{u}^k) \equiv\; & f(\mathbf{x}^k) + (\nabla f)_k^T(\Delta\mathbf{x}_0^k + \mathbf{L}_k\Delta\mathbf{u}^k) \\ & + \frac{1}{2}(\Delta\mathbf{x}_0^k + \mathbf{L}_k\Delta\mathbf{u}^k)^T(\nabla\nabla f)_k(\Delta\mathbf{x}_0^k + \mathbf{L}_k\Delta\mathbf{u}^k) \to \min_{\Delta\mathbf{u}^k} \end{aligned}$$

which can be cast in the form

$$\begin{aligned} \tilde{f}(\Delta\mathbf{u}^k) =\; & \frac{1}{2}(\Delta\mathbf{u}^k)^T\mathbf{L}_k^T(\nabla\nabla f)_k\mathbf{L}_k\Delta\mathbf{u}^k + \left[\mathbf{L}_k^T(\nabla\nabla f)_k\Delta\mathbf{x}_0^k + \mathbf{L}_k^T(\nabla f)_k\right]^T \Delta\mathbf{u}^k \\ & + \frac{1}{2}(\Delta\mathbf{x}_0^k)^T(\nabla\nabla f)_k\Delta\mathbf{x}_0^k + (\nabla f)_0^T\Delta\mathbf{x}_0^k + f(\mathbf{x}^k) \to \min_{\Delta\mathbf{u}^k} \end{aligned} \tag{139}$$

where $\tilde{f}(\Delta \mathbf{u}_k)$ is quadratic in $\Delta \mathbf{u}^k$, and has a minimum if the Hessian $(\nabla\nabla f)_k$ is positive-definite. Since this need not be the case, for f has been assumed only C^2-continuous, i.e., with continuous derivatives up to the second order, we may have to *stabilize* this Hessian, in order to render it positive-definite. The above procedure is called *sequential quadratic programming* with *Hessian stabilization*. Thus, if the lower bound l for the set of eigenvalues of $(\nabla\nabla f)_k$ is negative, which means that either $(\nabla\nabla f)_k$ is sign-indefinite or positive-definite but ill-conditioned, we add to it an isotropic matrix $\mu_k \mathbf{1}$, with μ_k, the stabilization factor, being a positive number. We then have

$$\mathbf{H}_k \leftarrow (\nabla\nabla f)_k + \mu_k \mathbf{1} \tag{140}$$

the value of μ_k being determined by means of the Gerschgorin Theorem, as described previously. We term this form of Hessian stabilization *Gerschgorin stabilization*.

In addition, the change takes place only in the quadratic term in $\Delta \mathbf{u}^k$. Therefore, $\tilde{f}(\Delta \mathbf{u}^k)$ can be written in the form

$$\begin{aligned}\tilde{f}(\Delta \mathbf{u}^k) &= \frac{1}{2}(\mathbf{L}_k \Delta \mathbf{u^k} + \mathbf{a}^k)^T \mathbf{H}_k (\mathbf{L}_k \Delta \mathbf{u^k} + \mathbf{a}^k)^T + b_k \\ &= \frac{1}{2}(\Delta \mathbf{u}^k)^T (\mathbf{L}_k)^T \mathbf{H}_k \mathbf{L}_k \Delta \mathbf{u}^k \\ &\quad + (\mathbf{a}^k)^T \mathbf{H}_k \mathbf{L}_k \Delta \mathbf{u}^k + \frac{1}{2}(\mathbf{a}^k)^T \mathbf{H}_k \mathbf{a}^k + b_k \end{aligned} \tag{141}$$

where $\mathbf{a}^k$ is found from

$$\mathbf{H}_k \mathbf{a}^k = (\nabla\nabla f)_k \Delta \mathbf{x}_0^k + (\nabla f)_k$$

while b_k is

$$b_k = \frac{1}{2}(\Delta \mathbf{x}_0^k)^T (\nabla\nabla f)_k \Delta \mathbf{x}_0^k + (\nabla f)_k^T \Delta \mathbf{x}_0^k + f(\mathbf{x}^k) - \frac{1}{2}(\mathbf{a}^k)^T \mathbf{H}_k \mathbf{a}^k$$

With $\mathbf{a}^k$ and $\mathbf{L}_k$ known, we next solve for the value of $\Delta \mathbf{u}^k$ rendering $\tilde{f}(\Delta \mathbf{u}^k)$ a minimum by first assuming

$$\mathbf{H}_k = \mathbf{V}_k^T \mathbf{V}_k$$

where $\mathbf{V}_k$ is the Cholesky factor of $\mathbf{H}_k$, this matrix thus playing the role of a weighting matrix. Then, $\tilde{f}(\Delta \mathbf{u}^k)$ can be minimized by simply computing the weighted least-square approximation of the system

$$\mathbf{V_k} \mathbf{L}_k \Delta \mathbf{u}^k = -\mathbf{V}_k \mathbf{a}^k$$

which consists of $n(> q)$ equations in q unknowns, the components of $\Delta \mathbf{u}^k$, the solution of this problem having been discussed previously.

7.3 Example 1: Powell's Function

A problem proposed by Powell (1969) is solved here. The problem is defined as: Find the minimum of a nonlinear objective function with five design variables, x_1, x_2, x_3, x_4, and x_5. The objective function f is defined as

$$f(\mathbf{x}) = e^{x_1 x_2 x_3 x_4 x_5} \tag{142}$$

The problem is subject to the nonlinear equality constraints

$$h_1 = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0 \tag{143a}$$

$$h_2 = x_2 x_3 + x_4 x_5 = 0 \tag{143b}$$

$$h_3 = x_1^3 + x_2^3 + 1 = 0 \tag{143c}$$

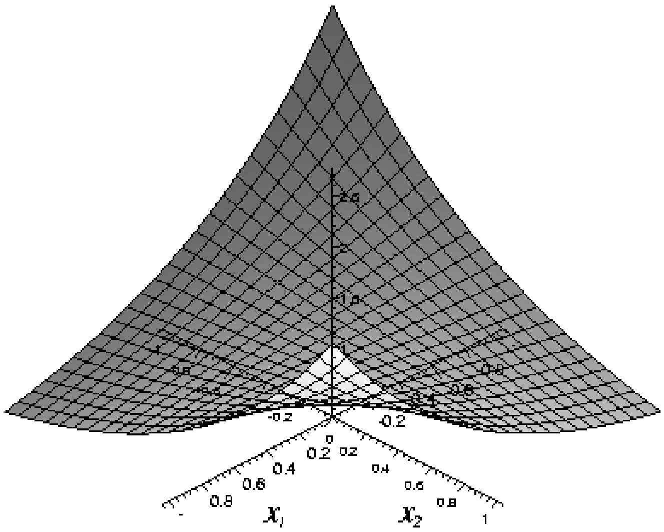

Figure 7: The bivariate exponential $e^{x_1 x_2}$

Here a word of caution is in order: while the exponential function e^x is convex—its second derivative with respect to x is positive everywhere—the bivariate exponential function $\mathbf{e}^{x_1 x_2}$ is not convex everywhere. As a consequence, its Hessian becomes sign-indefinite in a region of the x_1-x_2 plane. This statement is illustrated in Fig. 7. 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

Table 6: 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

of $\mathbb{R}^5$. Optimum solutions were obtained with two different algorithms, the corresponding results being listed in Table 6. 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 $\mathbf{x} = [-1, 2, -0.5, 1, 2]^T$ with tolerance of 10^{-6}. ODA 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*.

7.4 Example 2: The Equilibrium Configuration of an N-link Chain (Luenberger, 1984)

Shown in Fig. 8a 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 the said equilibrium configuration.

In this approach, angles θ_i are used to define the configuration of the chain. Angles 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. 9.

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

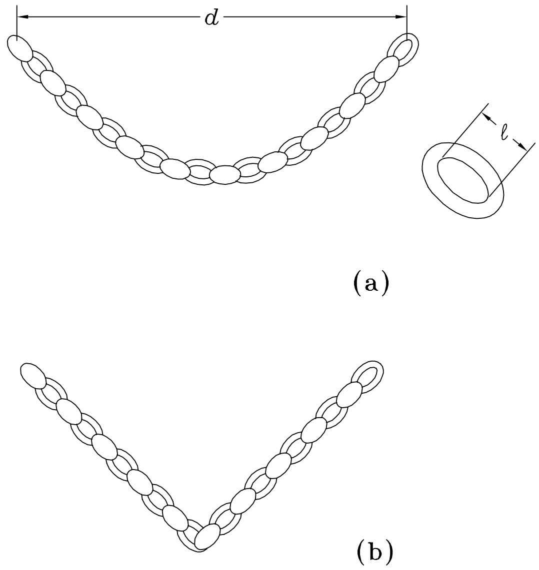

Figure 8: An N-link chain in: (a) its unknown equilibrium configuration; and (b) a configuration to be used as an initial guess

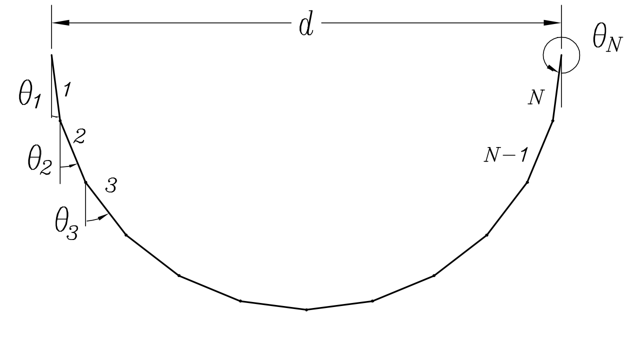

Figure 9: Definition of θ_i for the N-link chain

$$f(\theta_1,\, \theta_2\, \ldots,\, \theta_N) = -\left[\frac{1}{2}\cos\theta_1 + (\cos\theta_1 + \frac{1}{2}\cos\theta_2) + \ldots \right.$$
$$\left. + (\cos\theta_1 + \ldots + \cos\theta_{N-1} + \frac{1}{2}\cos\theta_{N-1})\right] \quad \rightarrow \quad \min_{\{\,\theta_i\,\}_1^N}$$

or, in compact form,

$$f(\theta_1,\, \theta_2\, \ldots,\, \theta_N) = -\frac{1}{2}\sum_{i=1}^{N}[2(N-i)+1]\cos\theta_i \quad \rightarrow \quad \min_{\{\,\theta_i\,\}_1^N} \tag{144}$$

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. 9. The constraints are

$$h_1 = \sum_{i=1}^{N}\cos\theta_i = 0 \tag{145a}$$
$$h_2 = \sum_{i=1}^{N}\sin\theta_i - \frac{d}{\ell} = 0 \tag{145b}$$

Under the assumption that the configuration is symmetric, only one half of the chain need be considered. The problem is, thus, simplified as

$$f(\theta_1,\, \theta_2\, \ldots,\, \theta_M) = -\left[\frac{1}{2}\cos\theta_1 + (\cos\theta_1 + \frac{1}{2}\cos\theta_2) + \ldots \right.$$
$$\left. + (\cos\theta_1 + \ldots + \cos\theta_{N-1} + \frac{1}{2}\cos\theta_{M-1})\right]$$
$$= -\frac{1}{2}\sum_{i=1}^{M}[2(M-i)+1]\cos\theta_i \quad \rightarrow \quad \min_{\{\,\cos\theta_i\,\}_1^M}$$

With, $M = N/2$, under the assumption that N is even, the two constraints then reduce to only one:

$$h = \sum_{i=1}^{M}\sin\theta_i - \frac{d}{2\ell} = 0 \tag{146a}$$

For $M = 2$, with $l = 0.5$, and $d = 1.25$, the contours of the objective function f and the constraint function h are displayed in Fig. 10. With the initial configuration given as in Fig. 8b, i.e, with $\theta_0 = [0.6751 \quad 0.6751]^T$, solutions were found by ODA as $\theta_1 = 0.4449$, $\theta_2 = 0.9607$, with $f_{min} = -1.6404$ in 52 iterations and 0.004273 s of CPU time. The optimum configuratrion found is displayed in Fig. 11.

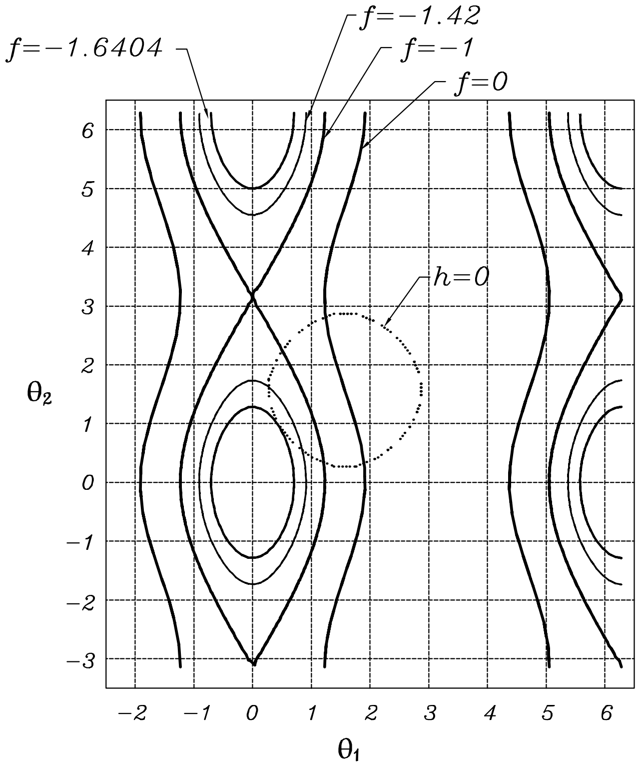

Figure 10: The Luenberger chain with $M = 2$

The same problem with a higher number of links, e.g., $M = 5$, i.e., with $N = 10$, is solved using the configuration of Fig. 8b as an initial guess. The equilibrium configuration of the chain is given in Table 7 with a comparison between ODA and *Matlab*.

With the same tolerance set at 0.0001, 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.(140), for a given value of $d/(2l)$.

8 Inequality-Constrained Problems

Throughout this tutorial we have shown applications of the ODA to optimization problems with *equality constraints*. Instances occur in which the constraints are *inequalities*, the said problems being thus called *inequality-constrained*. Nevertheless, inequality-constrained problems can be handled using the approach outlined

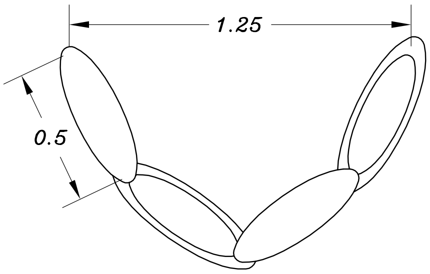

Figure 11: The optimum configuration of the Luenberger chain with $M = 2$

Table 7: 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

above 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: slack variables; interior or exterior penalty functions; etc. In this section, the methods of slack variables and interior penalty functions are outlined.

8.1 Slack Variables

Inequality constraints in an optimization problem are assumed to take the form

$$\mathbf{g}_k(\mathbf{x}) > 0, \quad k = 1, 2, \ldots, m$$

which is shorthand for

$$g_1(\mathbf{x}) > 0$$

Table 8: Luenberger's chain with $N = 10$

	Matlab	ODA
θ_1	0.0455	0.0455
θ_2	0.0585	0.0585
θ_3	0.0818	0.0818
θ_4	0.1357	0.1357
θ_5	0.3888	0.3888
θ_6	2.7527	2.7527
θ_7	3.0059	3.0059
θ_8	3.0598	3.0598
θ_9	3.0831	3.0831
θ_{10}	3.0961	3.0961
f_{min}	-24.8598	-24.8598
Iterations	38	208
CPU time (s)	5.0442	0.742537

$$\begin{aligned} g_2(\mathbf{x}) &> 0 \\ &\vdots \\ g_m(\mathbf{x}) &> 0 \end{aligned} \tag{147}$$

Here, note that the vector inequality of expression (147) means that each component of vector $\mathbf{g}(\mathbf{x})$ obeys an inequality constraint of the form "> 0," and should not be interpreted as a "vector inequality", which is not possible, since vectors do not form *ordered sets*.

We introduce now the *slack variables* $s_1, s_2, \cdots, s_m$, such that

$$\mathbf{h}(\mathbf{x}) \equiv \begin{bmatrix} g_1 - s_1^2 \\ g_2 - s_2^2 \\ \vdots \\ g_l - s_m^2 \end{bmatrix} = \mathbf{0} \tag{148}$$

thereby converting the inequality constraints into equality constraints. Notice that the slack variables are now additional design variables, the dimension of the design space being correspondingly increased. In consequence, the design vector is now

$$\mathbf{x} = [\, x_1 \quad x_2 \quad \cdots \quad x_n \quad s_1 \quad s_2 \quad \cdots \quad s_m \,]^T \tag{149}$$

Table 9: Input-output pairs

ψ (deg.)	70	80	90	100	110	130	140
ϕ (deg.)	40	45	50	58	64	74	80

8.1.1 Example: Minimization of the Design Error of a Four-Bar Linkage with an Input Crank

The aim of this example is to determine the value of the *Freudenstein parameters* k_1, k_2 and k_3 of a four-bar function-generating linkage, like that shown in Fig. 2, such that the norm of the *design error* defined in eq.(23) attains its minimum. The set of input-output pairs $\{\psi_i, \phi_i\}_1^q$ is shown in Table 9. Extra constraints are imposed here to obtain a full mobility in the input link, i.e., an input crank. The *conditions for full mobility* are (Liu and Angeles, 1992)

$$g_1(\mathbf{x}) = (k_2 + k_1)^2 - (k_3 - k_4)^2 < 0 \tag{150a}$$

$$g_2(\mathbf{x}) = (k_2 - k_1)^2 - (k_3 + k_4)^2 < 0 \tag{150b}$$

By introducing two slack-variables s_1 and s_2, the inequality constraints are converted into equality constraints, i.e.,

$$h_1(\mathbf{x}) = (k_2 + k_1)^2 - (k_3 - k_4)^2 + s_1 = 0 \tag{151a}$$

$$h_2(\mathbf{x}) = (k_2 - k_1)^2 - (k_3 + k_4)^2 + s_2 = 0 \tag{151b}$$

The design vector $\mathbf{x}$ thus becomes $\mathbf{x} = [k_1\ k_2\ k_3\ k_4\ s_1\ s_2]^T$. From the initial guess $\mathbf{x}_0 = [0.28\ 0.74\ 0.12\ 1.69\ 1.2]^T$, the solution was found to be $\mathbf{x}_{opt} = [0.3248\ \ 0.5875\ \ -0.009725\ 1.556\ 0.2415]^T$, and the corresponding link lengths are $a_1 = 1$, $a_2 = 1.702$, $a_3 = 103.4$ and $a_4 = 102.8$. The Euclidean norm of the minimum design error is 5×10^{-2}.

8.2 Interior 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: either the solution is approached *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.

Given an objective function $f(\mathbf{x})$ subject to inequality constraints, as defined in eq.(147), a sequence of interior penalty functions $\{\phi_k\}_1^\nu$ is constructed as

$$\phi_k(\mathbf{x}; r_k) \equiv f(\mathbf{x}) + r_k \sum_{i=1}^{\nu} \frac{1}{h_i(\mathbf{x})} \quad k = 1,\, 2,\, \cdots,\, l \tag{152}$$

where all r_k's are positive and observe a decreasing order, i.e.,

$$r_1 > r_2 > r_3 \cdots > r_\nu \tag{153}$$

Now, a sequence of unconstrained optimization problems is defined:

$$\phi_k(\mathbf{x}; r_k) \equiv f(\mathbf{x}) + r_k \sum_{i=1}^{\nu} \frac{1}{h_i(\mathbf{x})} \quad \rightarrow \quad \min_{\mathbf{x}}, \quad k = 1,\, 2,\, \cdots,\, \nu \tag{154}$$

Let $\mathbf{x}_o^1$, $\mathbf{x}_o^2$, …, $\mathbf{x}_o^\nu$ be the sequence of corresponding unconstrained optima. Next, these optima are interpolated to a vector function $\mathbf{x}_o(r)$:

$$\mathbf{x}_o(r) \equiv \mathbf{c}_0 + \sum_{1}^{\nu-1} \mathbf{c}_k r^{k/2} \tag{155}$$

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.(155) is written for $r = r_i$, with $i = 1,\, 2,\, \ldots,\, \nu$:

$$\mathbf{x}_o(r_i) \equiv \mathbf{c}_0 + \mathbf{c}_1 r_i^{1/2} + \mathbf{c}_2 r_i^{2/2} + \cdots + \mathbf{c}_{\nu-1} r_i^{(\nu-1)/2} \tag{156}$$

or

$$\mathbf{x}_o(r_i) \equiv [\, \mathbf{c}_0 \quad \mathbf{c}_1 \quad \cdots \quad \mathbf{c}_{\nu-1} \,] \begin{bmatrix} 1 \\ r_i^{1/2} \\ \vdots \\ r_i^{(\nu-1)/2} \end{bmatrix}, \quad i = 1,\, 2,\, \ldots,\, \nu$$

In the next step, we regroup all ν vector equations above to produce a matrix equation. To this end, we define the matrices

$$\mathbf{R} \equiv \begin{bmatrix} 1 & 1 & \cdots & 1 \\ r_1^{1/2} & r_2^{1/2} & \cdots & r_\nu^{1/2} \\ \vdots & \vdots & \ddots & \vdots \\ r_1^{(\nu-1)/2} & r_2^{(\nu-1)/2} & \cdots & r_\nu^{(\nu-1)/2} \end{bmatrix} \tag{157a}$$

$$\mathbf{X}_o \equiv [\, \mathbf{x}_o(r_1) \quad \mathbf{x}_o(r_2) \quad \mathbf{x}_o(r_\nu) \,] \tag{157b}$$

$$\mathbf{C} \equiv [\, \mathbf{c}_0 \quad \mathbf{c}_1 \quad \cdots \quad \mathbf{c}_{\nu-1} \,] \tag{157c}$$

Thus, the ν vector equations (156) become, in matrix form,

$$\mathbf{CR} = \mathbf{X}_o \tag{158a}$$

whence,

$$\mathbf{C} = \mathbf{X}_o\mathbf{R}^{-1} \tag{158b}$$

or, if eq.(158a) 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

$$\mathbf{C}^T = \mathbf{R}^{-T}\mathbf{X}_o^T \tag{158c}$$

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

$$\mathbf{x}_{\text{opt}} = \lim_{r\to 0} \mathbf{x}_{\text{opt}}(r)$$

i.e.,

$$\mathbf{x}_{\text{opt}} = \mathbf{c}_0 \tag{159}$$

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 $\mathbf{c}_0$.

8.2.1 Example: A Two-dimensional Optimization Problem Subject to Inequality Constraints

Consider an optimization problem with an objective function defined as

$$f = x^2 + 2y^2 \quad \to \quad \min_{x,y} \tag{160}$$

subject to inequality constraints

$$g_1 \equiv -x \leq 0 \tag{161a}$$
$$g_2 \equiv -y \leq 0 \tag{161b}$$
$$g_3 \equiv 1 - x - y \leq 0 \tag{161c}$$

From a sketch of the constraints and the contours of the objective function in the

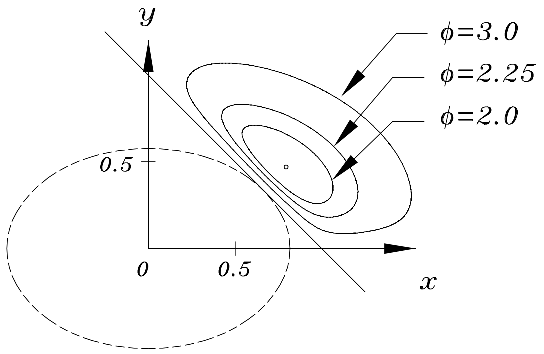

Figure 12: Isocontours of the penalty function with $r_1 = 0.1$

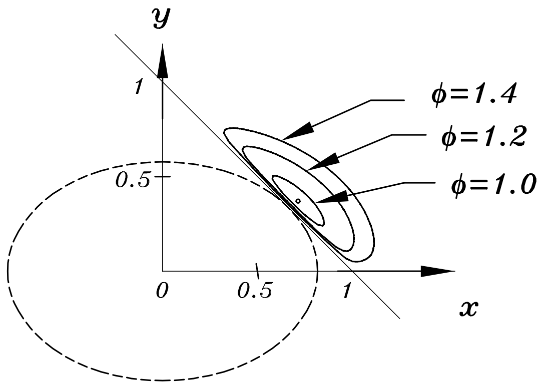

Figure 13: Isocontours of the penalty function with $r_2 = 0.01$

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

$$x_{\text{opt}} = \frac{2}{3}, \quad y_{\text{opt}} = \frac{1}{3}$$

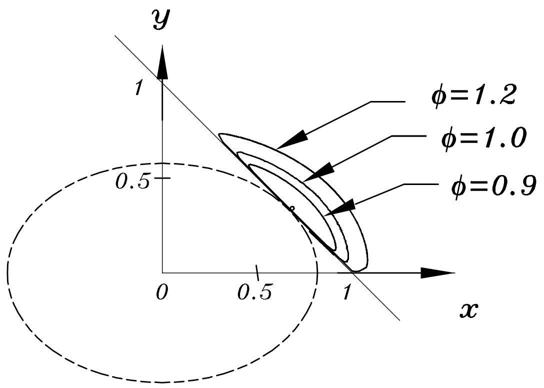

Figure 14: Isocontours of the penalty function with $r_3 = 0.001$

Below, we demonstrate 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. 12–14. In those figures, the isocontour of the objective function f that corresponds to the constrained minimum is indicated with a dash 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 as

$$\mathbf{x}_o(r_1) = \begin{bmatrix} 0.7941 \\ 0.4704 \end{bmatrix}, \quad \mathbf{x}_o(r_2) = \begin{bmatrix} 0.7140 \\ 0.3703 \end{bmatrix}, \quad \mathbf{x}_o(r_3) = \begin{bmatrix} 0.6836 \\ 0.3434 \end{bmatrix} \tag{162}$$

We now fit the values of $\{\mathbf{x}_o(r_k)\}_1^3$ to the function

$$\mathbf{x}_o(r_k) = \mathbf{c}_0 + \mathbf{c}_1 r_k^{1/2} + \mathbf{c}_2 r_k, \quad k = 1, 2, 3$$

We thus have

$$\mathbf{R} = \begin{bmatrix} 1 & 1 & 1 \\ 0.3163 & 0.1000 & 0.03163 \\ 0.1000 & 0.0100 & 0.00100 \end{bmatrix}, \quad \mathbf{X}_o = \begin{bmatrix} 0.7941 & 0.7140 & 0.6836 \\ 0.4704 & 0.3703 & 0.3434 \end{bmatrix} \tag{163}$$

The coefficient matrix $\mathbf{C}$ is thus found to be

$$\mathbf{C} = \mathbf{X}_o\mathbf{R}^{-1} = \begin{bmatrix} 0.6687 & 0.4790 & -0.2605 \\ 0.3317 & 0.3612 & 0.2443 \end{bmatrix} \tag{164}$$

Therefore,

$$\mathbf{x}_{\text{opt}} = \mathbf{c}_0 = \begin{bmatrix} 0.6687 \\ 0.3317 \end{bmatrix} \tag{165}$$

which yields the optimum with two significant digits of accuracy.

9 Computer Implementation Using the Software Package ODA

ODA is a general-purpose software package for optimization problems. The package was originally developed as *QUADMIN* (Angeles and Ma, 1989). New functions, such as those needed in solving constrained optimization problems with arbitrary objective function have been added to QUADMIN. The new package, ODA, is no longer limited to finding the minimum of objective functions that are quadratic in a nonlinear vector function of the design variables.

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 $\mathbf{W}$, and 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*.

References

Angeles, J., 1982, *Spatial Kinematic Chains*, Springer-Verlag, New York.

Angeles, J., Anderson, K. and Gosselin, C., 1990, "Constrained design optimization using orthogonal decomposition", *ASME J. Mechanisms, Transmissions, and Auto. in Design*, Vol. 112, No. 2, pp. 255-256.

Bertsekas, D. P., 1995, *Nonlinear Programming*, Athena Scientific, Belmont, Massachusetts.

Betts, J. T., 1980, "A compact algorithm for computing the stationary point of a quadratic function subject to linear constraints", *ACM Trans. on Math. Software*, Vol. 63, pp. 391-397.

Broyden, C. G., 1970, "The convergence of a class of double-rank minimization algorithm", *J. Inst. Maths. Applics.*, Vol. 6, pp. 76-90.

Fox, R. L. and Gupta, K. C., 1973, "Optimization technology as applied to mechanism design", *ASME Journal of Engineering for Industry*, pp. 657-661.

Freudenstein, F., 1955, "Approximate synthesis of four-bar linkages", *Trans. ASME*, Vol. 77, pp. 853-861.

Goldfarb, D., 1970, “A family of variable metric updates derived by variational means”, *Mathematics of Computing*, Vol. 24, pp. 23-26.

Golub, G. H. and Van Loan, C. F., 1983,*Matrix Computations*, The Johns Hopkins University Press, Baltimore.

Hartenberg R. S. and Denavit, J., 1964, *Kinematic Synthesis of Linkages*, McGraw-Hill Book Co., New York.

Kahaner D., Moler C. and Nash S., 1989, *Numerical methods and software*, Prentice Hall, Englewood Cliffs, NJ.

Lalee, M., Nocedal, J. and Plantenga, 1998, “On the implementation of an algorithm for large-scale equality constrained optimization”, *SIAM J. Optim.*, Vol. 8, pp. 682-706.

Liu, Z. and Angeles, J., 1992, “Least-square optimization of planar and spherical four-bar function generator under mobility constraints”, *ASME, J. of Mech. Des.*, Vol. 114, pp. 569-573.

Mangasarian, O. L., 1972, “Techniques of optimization”, *Trans. ASME Journal of Engineering for Industry*, Vol. 93, pp. 365-371.

Murray, W., 1997, “Sequential quadratic programming methods for large problems”, *Comput. Optim. Appl.*, Vol. 7, pp. 127-142.

Powell, M. J. D., 1969, “A method for Nonlinear Constraints in Minimizing Problems”, in Fletcher, R. (Ed.), *Optmization*, Academic Press., N. Y., pp. 283-298.

Rao, S. S., 1996, *Engineering Optimization*, 3rd. Edition, John Wiley and Sons, Inc., New York.

Rosenbrock, H. H., 1960, “An automatic method for finding the greatest or the least value of a function”, *Computer Journal*, Vol. 3. No. 3, pp. 175-184.

Schittkowski, K., 1985, "A unified outline of nonlinear programming algorithms", *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 107, pp. 449-453.

Seireg, A., 1972, "A survey of optimization of mechanical design", *ASME Journal Engineering for Industry*, Vol. 94, pp. 495-499.

Shanno, D. F., 1970, "Conditioning of quasi-Newton methods for function minimization", *Mathematics of Computing*, Vol. 24, pp. 647-656.

Stewart, G. W., 1973, *Introduction to Matrix Computations, Academic Press*, New York.

Strang, G., 1988, *Linear Algebra*, Third Edition, Harcourt Brace Jovanovich College Publishers, Fort Worth.

Teng, C. and Angeles, J., 1999, "A sequential-quadratic-programming algorithm using orthogonal decomposition with Gerschgorin stabilization", *1999 ASME Design Engineering Technical Conferences.*

Varga, R. S., 1962, *Matrix Iterative Analysis*, Prentice-Hall Inc., Englewood Cliffs, N. J..

Zhou, J. and Mayne, R. W., 1985, "Monotonicity analysis and recursive quadratic programming in constrained optimization", *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 107, pp. 459-462.