OPTIMACION DE SISTEMAS FISICOS

DIRECTORIO DE PROFESORES 1981

DR. JORGE ANGELES ALVAREZ
SUBJEFE DEL AREA MECANICA ELECTRICA
D. E. P. F. I., UNAM
CIUDAD UNIVERSITARIA
MEXICO 20, D.F.
TEL: 550.52.15 ext. 4493

DR. EDUARDO ARRIOLA VALDES
JEFE DEL DEPARTAMENTO DE METODOLOGIA
GERENCIA DE ESTUDIOS ELECTRICOS
COMISION FEDERAL DE ELECTRICIDAD
MELCHOR OCAMPO No. 469 piso 1
MEXICO 17, D.F.
TEL: 511. 00. 99

DR. ENRIQUE CHICUREL UZIEL
JEFE DE LA SECCION DE INGENIERIA MECANICA
D. E. P. F. I., UNAM
CIUDAD UNIVERSITARIA
MEXICO 20, D.F.
TEL: 550.52.15 ext. 4493

M. EN C. SUSANA GOMEZ GOMEZ
INVESTIGADORA ASOCIADA B
INSTITUTO DE INVESTIGACION EN MATEMATICAS
APLICADAS Y EN SISTEMAS, UNAM
CIUDAD UNIVERSITARIA
MEXICO 20, D.F.
TEL: 550.52. 15 ext. 4572 6 4578

DR. MARCO ANTONIO MURRAY-LASSO
PROFESOR DE MEDIO TIEMPO
D. E. P. F. I., UNAM
CIUDAD UNIVERSITARIA
MEXICO 20, D.F.
TEL: 550.52.15 ext. 4493

DR. ALEJANDRO VELASCO LEVY
JEFE DEL PROYECTO DEL LABORATORIO RAMSES
INSTITUTO DE INVESTIGACION EN MATEMATICAS
APLICADAS Y EN SISTEMAS, UNAM
CIUDAD UNIVERSITARIA
MEXICO 20, D.F.
TEL: 550.52. 15 ext. 4565 y 548.54.65

'pmc.

OPTIMACION [ ]TEMAS FISICOS
(2 al 6 d[ ].rzo de 1981)

F E C H A	HORARIO		T E M A S	P R O F E S O R E S
2 de marzo	9 a 17	1.	INTRODUCCION Antecedentes matemáticos y numéricos de las técnicas de optimación	DR. JORGE ANGELES ALVAREZ
3 de marzo	9 a 11	2.	CONDICIONES DE OPTIMALIDAD	M. EN C. SUSANA GOMEZ GOMEZ
	11:30 a 17	2a.	METODOS DE OPTIMACION SIN RESTRICCIONES	M. EN C. SUSANA GOMEZ GOMEZ
4 de marzo	9 a 11		METODOS DE OPTIMACION SIN RESTRICCIONES	M. EN C. SUSANA GOMEZ GOMEZ
	11:30 a 17	3.	METODOS DE OPTIMACION CON RESTRICCIONES	DR. ALEJANDRO VELASCO LEVY
5 de marzo	9 a 17		METODOS DE OPTIMACION CON RESTRICCIONES	DR. ALEJANDRO VELASCO LEVY
6 de marzo		4 y 5	PROBLEMAS ASOCIADOS A SISTEMAS MECANICOS TERMICOS, QUIMICOS, ELECTRICOS Y COMBINACIONES DE ESTOS. PAQUETES DE PROGRAMAS.	
	9 a 10		OPTIMACION DE ELEMENTOS DE MAQUINAS	DR. ENRIQUE CHICUREL UZIEL
	10 a 11		OPTIMACION DE TRENES INTERCAMBIADORES DE COLOR	M. EN C. SUSANA GOMEZ GOMEZ
	11:30 a 13:30		OPTIMACION DE CIRCUITOS ELECTRICOS	DR. MARCO ANTONIO MURRAY-LASSO
	15 a 16		OPTIMACION DE SISTEMAS ELECTRICOS DE POTENCIA	DR. EDUARDO ARRIOLA VALDES
	16 a 17		OPTIMACION DE MECANISMOS	DR. JORGE ANGELES ALVAREZ
			C L A U S U R A	

'pmc.

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

OPTIMACION DE SISTEMAS FISICOS

ANTECEDENTES MATEMATICOS Y NUMERICOS DE LAS TECNICAS DE OPTIMACION

DR. JORGE ANGELES ALVAREZ

MARZO, 1981

1. MATHEMATICAL PRELIMINARIES

1.0 INTRODUCTION. Some relevant mathematical results are collected in this chapter. These results find a wide application within the realm of analysis, synthesis and optimization of mechanisms. Often, rigorous proofs are not provided; however a reference list is given at the end of the chapter, where the interested reader can find the required details.

1.1. VECTOR SPACE, LINEAR DEPENDENCE AND BASIS OF A VECTOR SPACE.

A vector space, also called a linear space, over a field F (1.1)* , is a set V of objects, called vectors, having the following properties:

a) To each pair $\{\underset{\sim}{x}, \underset{\sim}{y}\}$ of vectors from the set, there corresponds one (and only one) vector, denoted $\underset{\sim}{x} + \underset{\sim}{y}$, also from V, called "the addition of $\underset{\sim}{x}$ and $\underset{\sim}{y}$" such that

   i) This addition is commutative, i.e.

$$\underset{\sim}{x} + \underset{\sim}{y} = \underset{\sim}{y} + \underset{\sim}{x}$$

   ii) It is associative, i.e., for any element $\underset{\sim}{z}$ of V,

$$\underset{\sim}{x} + (\underset{\sim}{y} + \underset{\sim}{z}) = (\underset{\sim}{x} + \underset{\sim}{y}) + \underset{\sim}{z}$$

   iii) There exists in V a unique vector $\underset{\sim}{0}$, called "the zero of V", such that, for any $\underset{\sim}{x} \in V$,

$$\underset{\sim}{x} + \underset{\sim}{0} = \underset{\sim}{x}$$

   iv) To each vector $\underset{\sim}{x} \in V$, there corresponds a unique vector $-\underset{\sim}{x}$, also in V, such that

$$\underset{\sim}{x} + (-\underset{\sim}{x}) = \underset{\sim}{0}$$

---

* Numbers in brackets designate references at the end of each chapter.

The said set of vectors is linearly independent (ℓ. i.) if $\underset{\sim}{c}$ equals zero implies that all α's are zero as well. Otherwise, the set is said to be linearly dependent (ℓ. d.)

Example 1.1.4 The set containing only one nonzero vector, $\{\underset{\sim}{x}\}$, is ℓ.i.

Example 1.1.5 The set containing only two vectors, one of which is the origin, $\{\underset{\sim}{x},\underset{\sim}{0}\}$, is ℓ.d.

The set of vectors $\{\underset{\sim}{x}_1,\underset{\sim}{x}_2,\ldots,\underset{\sim}{x}_n\} \subset V$ spans V if and only if every vector $\underset{\sim}{v} \in V$ can be expressed as a linear combination of the vectors of the set.

A set of vectors $B = \{\underset{\sim}{x}_1,\underset{\sim}{x}_2,\ldots,\underset{\sim}{x}_n\} \subset V$ is a basis for V if and only if:

i) B is linearly independent, and

ii) B spans V

All bases of a given space V contain the same number of vectors. Thus, if B is a basis for V, the number n of elements of B is the dimension of V (abreviated: n=dim V)

Example 1.1.6 In 3-dimensional Euclidean space the unit vectors $\{\underset{\sim}{i}, \underset{\sim}{j}\}$ lying parallel to the X and Y coordinate axes span the vectors in the X-Y plane, but do not span the vectors in the physical three-dimensional space.

Exercise 1.1.1 Prove that the set B given above is a basis for V if and only if each vector in V can be expressed as a unique linear combination of the elements of B.

1.2 LINEAR TRANSFORMATION AND ITS MATRIX REPRESENTATION

Henceforth, only finite-dimensional vector spaces will be dealt with and, when necessary, the dimension of the space will be indicated as an exponent of the space, i.e., V^n means dim V=n.

A transformation T, from an m-dimensional vector space U, to an n-dimensional vector space V is a rule which establishes a correspondence between an element of U and a unique element of V. It is represented as:

that, for all distinct $\underset{\sim}{u}_1$ and $\underset{\sim}{u}_2$, $\underset{\sim}{T}(\underset{\sim}{u}_1)$ and $\underset{\sim}{T}(\underset{\sim}{u}_2)$ are also distinct, $\underset{\sim}{T}$ is said to be one-to-one. If $\underset{\sim}{T}$ is onto and one-to-one, it is said to be invertible.

If $\underset{\sim}{T}$ is invertible, to each $\underset{\sim}{v} \in V$ there corresponds a unique $\underset{\sim}{u} \in U$ such that $\underset{\sim}{v} = \underset{\sim}{T}(\underset{\sim}{u})$, so one can define a mapping $\underset{\sim}{T}^{-1}: V \to U$ such that

$$\underset{\sim}{u} = \underset{\sim}{T}^{-1}(\underset{\sim}{v}) \tag{1.2.4}$$

$\underset{\sim}{T}^{-1}$ is called the "inverse" of $\underset{\sim}{T}$.

Exercise 1.2.2 Let $\underset{\sim}{P}$ be the projection of the three-dimensional Euclidean space onto a plane, say, the X-Y plane. Thus, $\underset{\sim}{v} = \underset{\sim}{P}(\underset{\sim}{u})$ is such that the vector with components (x, y, z), is mapped into the vector with components (x, y, 0).

i) Is $\underset{\sim}{P}$ a linear transformation?

ii) Is $\underset{\sim}{P}$ onto?, one-to-one?, invertible?

A very important fact concerning linear transformations of finite dimensional vector spaces is contained in the following result:

Let $\underset{\sim}{L}$ be a linear transformation from U^m to V^n. Let B_u and B_v be bases for U^m and V^n, respectively. Then clearly, for each $\underset{\sim}{u}_i \in B_u$ its image $\underset{\sim}{L}(\underset{\sim}{u}_i) \in V$ can be expressed as a linear combination of the $\underset{\sim}{v}_k$'s in B_v. Thus

$$\underset{\sim}{L}(\underset{\sim}{u}_i) = \alpha_{1i}\underset{\sim}{v}_1 + \alpha_{2i}\underset{\sim}{v}_2 + \ldots + \alpha_{ni}\underset{\sim}{v}_n \tag{1.2.5}$$

Consequently, to represent the images of the m vectors of B_u, mn scalars like those appearing in (1.2.5) are required. These scalars can be arranged in the following manner:

$$[\underset{\sim}{A}] = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \ldots & \alpha_{1m} \\ \alpha_{21} & \alpha_{22} & \ldots & \alpha_{2m} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \alpha_{n1} & \alpha_{n2} & \ldots & \alpha_{nm} \end{bmatrix} \tag{1.2.6}$$

1.3 RANGE AND NULL SPACE OF A LINEAR TRANSFORMATION

As stated in Section 1.2, the set of vectors $\underset{\sim}{v} \in V$ for which there is at least one $\underset{\sim}{u} \in U$ such that $\underset{\sim}{v} = L(\underset{\sim}{u})$ is called "the range of L" and is represented as $R(L)$, i.e. $R(L) = \{v=L(u) : u \in U\}$.

The set of vectors $\underset{\sim}{u}_0 \in U$ for which $L(\underset{\sim}{u}_0) = \underset{\sim}{0} \in V$ is called "the null space of L" and is represented as $N(L)$, i.e. $N(L) = \{u_0 : L(u_0)=0\}$.

It is a simple matter to show that $R(L)$ and $N(L)$ are subspaces of V and U, respectively*.

The dimensions of dom(L), R(L) and N(L) are not independent, but they are related (see (1.2)):

$$\dim \text{dom}(L) = \dim R(L) + \dim N(L) \qquad (1.3.1)$$

Example 1.3.1 In considering the projection of Exercise 1.2.1, U is E^3 and thus R(P) is the X-Y plane, N(P) is the Z axis, hence of dimension 1. The X-Y plane is two-dimensional and dom(L) is three-dimensional, hence (1.3.1) holds.

Exercise 1.3.1 Describe the range and the null space of the reflection of Example 1.2.1 and verify that eq. (1.3.1) holds true.

1.4 EIGENVALUES AND EIGENVECTORS OF A LINEAR TRANSFORMATION

Let L be a linear transformation of V into itself (such an L is called an "endomorphism"). In general, the image $L(v)$ of an element v of V is linearly independent with v, but if it happens that a nonzero vector v and its image under L are linearly dependent, i.e. if

$$L(v) = \lambda v \qquad (1.4.1)$$

---

* The proof of this statement can be found in any of the books listed in the reference at the end of this chapter.

"the geometric multiplicity of λ_i".

Exercise 1.4.1 Show that the geometric multiplicity of a particular eigenvalue cannot be greater than its algebraic multiplicity.

A Hermitian matrix is one which equals its transpose conjugate. If a matrix equals the negative of its transpose conjugate, it is said to be skew Hermitian. For Hermitian matrices we have the very important result:

THEOREM 1.4.1 The eigenvalues of a Hermitian matrix are real and its eigenvectors are mutually orthogonal (i.e. the inner product, which is discussed in detail in Sec. 1.8, of two distinct eigenvectors, is zero).

The proof of the foregoing theorem is very widely known and is not presented here. The reader can find a proof in any of the books listed at the end of the chapter.

1.5 CHANGE OF BASIS

Given a vector $\underline{v}$, its representation $(v_1, v_2, \ldots, v_n)^T$ referred to a basis $B = (\underline{\beta}_1, \underline{\beta}_2, \ldots, \underline{\beta}_n)$, is defined as the ordered set of scalars that produce $\underline{v}$ as a linear combination of the vectors of B. Thus, $\underline{v}$ can be expressed as

$$\underline{v} = v_1\underline{\beta}_1 + v_2\underline{\beta}_2 + \ldots + v_n\underline{\beta}_n \qquad (1.5.1)$$

A vector $\underline{v}$ and its representation, though isomorphic* to each other, are essentially different entities. In fact, $\underline{v}$ is an abstract algebraic entity satisfying properties a) and b) of Section 1.1, whereas its representation is an array of numbers. Similarly, a linear transformation, $\underline{L}$, and its representation, $(\underline{L})_B$, are essentially different entities. A question that could arise naturally is: Given the representations $(\underline{v})_B$ and $(\underline{L})_B$ of $\underline{v}$ and L, respectively, referred to the basis B, what are the corresponding

---

* Two sets are isomorphic to each other if similar operations can be defined on their elements.

or, equivalently,

$$(\underline{v})_C = (\underline{A})_B^{-1}(\underline{v})_B \qquad (1.5.8)$$

Now, assuming that $\underline{w}$ is the image of $\underline{v}$ under $\underline{L}$,

$$(\underline{w})_B = (\underline{L})_B (\underline{v})_B \qquad (1.5.9)$$

or, referring eq. (1.5.9) to the basis C, instead,

$$(\underline{w})_C = (\underline{L})_C (\underline{v})_C \qquad (1.5.10)$$

Applying the relationship (1.5.8) to vector $\underline{w}$ and introducing it into eq. (1.5.10),

$$(\underline{A}^{-1})_B (\underline{w})_B = (\underline{L})_C (\underline{A})_B^{-1}(\underline{v})_B$$

from which the next relationship readily follows

$$(\underline{w})_B = (\underline{A})_B (\underline{L})_C (\underline{A})_B^{-1} (\underline{v})_B \qquad (1.5.11)$$

Finally, comparing (1.5.9) with (1.5.11),

$$(\underline{L})_B = (\underline{A})_B (\underline{L})_C (\underline{A})_B$$

or, equivalently,

$$(\underline{L})_C = (\underline{A})_B^{-1} (\underline{L})_B (\underline{A})_B \qquad (1.5.12)$$

Relationships (1.5.8) and (1.5.12) are the answers to the question posed at the beginning of this Section. The right hand side of (1.5.12) is a similarity transformation of $(\underline{L})_B$

Exercise 1.5.1 Show that, under a similarity transformation, the characteristic polynomial of a matrix remains invariant.

Exercise 1.5.2 The trace of a matrix is defined as the sum of the elements on its diagonal. Show that the trace of a matrix remains invariant under a similarity transformation Hint: Show first that, if $\underline{A}$, $\underline{B}$ and $\underline{C}$ are nxn matrices,

$$\mathrm{Tr}(\underline{A}\underline{B}\underline{C}) = \mathrm{Tr}(\underline{B}\underline{C}\underline{A}).$$

where $\underline{e}_i^T$ is the transpose of $\underline{e}_i$ ($\underline{e}_i$ being a column vector, $\underline{e}_i^T$ is a row vector). The whole set of equations (1.6.6), for all i and all j can then be written as

$$\underline{Q}^T\underline{Q} = \underline{I} \tag{1.6.7}$$

where $\underline{I}$ is the matrix with unity on its diagonal and zeros elsewhere. Eq. (1.6.7) states a very important fact about $\underline{Q}$, namely, that it is an orthogonal matrix. Summarizing, a symmetric nxn matrix $\underline{A}$ can be diagonalized via a similarity transformation, the columns of whose matrix are the eigenvectors of $\underline{A}$

The eigenvalue problem stated in (1.6.1) is solved by first finding the eigenvalues $(\lambda_i)_1^n$. These values are found from the following procedure: Write eq. (1.6.1) in the form

$$(\underline{A} - \lambda_i\underline{I})\underline{e}_i = \underline{0} \tag{1.6.8}$$

This equation states that the set $(\underline{e}_i)_1^n$ lies in the null space of $\underline{A} - \lambda_i\underline{I}$. For this matrix to have nonzero vectors in its null space, its determinant should vanish, i.e.

$$\det(\underline{A}-\lambda_i\underline{I}) \equiv P(\lambda_i) = 0 \tag{1.6.9}$$

whose left hand side is its characteristic polynomial, which was introduced in section 1.4. This equation thus contains n roots, some of which could be repeated.

A very useful result is next summarized, though not proved.

THEOREM (Cayley-Hamilton). A square matrix satisfies its own characteristic equation, i.e. if $P(\lambda_i)$ is its characteristic polynomial, then

$$P(\underline{A}) = \underline{0} \tag{1.6.10}$$

A proof of this teorem can be found either in (1.3, pp. 148-150) or in (1.4, pp. 112-115)

ii) $\phi(\underline{v},\underline{u})$ is the complex conjugate of $\phi(\underline{u},\underline{v})$, i.e.

$$\phi(\underline{v},\underline{u}) = \bar{\phi}(\underline{u},\underline{v}) \qquad (1.7.1e)$$

The foregoing properties of conjugate bilinear forms suggest that one possible way of constructing a bilinear form is as follows:

Let

$$\phi(\underline{u},\underline{v}) = \underline{u}^*\underline{A}\underline{v} \qquad (1.7.2)$$

Exercise 1.7.1 Prove that definition (1.7.2) satisfies properties (1.7.1)

If, in (1.7.1), $\underline{v} = \underline{u}$, the bilinear form becomes the quadratic form

$$\phi(\underline{u}) = \underline{u}^*\underline{A}\,\underline{u} \qquad (1.7.3)$$

It will be shown that the bilinear form (1.7.2) defines a scalar product for a vector space under certain conditions on $\underline{A}$.

Definition: A scalar product, $p(\underline{u},\underline{v})$, of two elements for a vector space U is a complex number with the following properties:

i) It is Hermitian symmetric:

$$p(\underline{u},\underline{v}) = \bar{p}(\underline{v},\underline{u}) \qquad (1.7.4a)$$

ii) It is conjugate linear in both $\underline{u}$ and $\underline{v}$:

$$p(\underline{u}_1+\underline{u}_2,\underline{v}) = p(\underline{u}_1,\underline{v}) + p(\underline{u}_2,\underline{v}) \qquad (1.7.4b)$$

$$p(\underline{u},\underline{v}_1+\underline{v}_2) = p(\underline{u},\underline{v}_1) + p(\underline{u},\underline{v}_2) \qquad (1.7.4c)$$

$$p(\alpha\underline{u},\underline{v}) = \alpha p(\underline{u},\underline{v}) \qquad (1.7.4d)^*$$

$$p(\underline{u},\beta\underline{v}) = \bar{\beta}p(\underline{u},\underline{v}) \qquad (1.7.4e)$$

iii) It is real and positive definite:

$$p(\underline{u},\underline{u})>0, \text{ for } \underline{u} \neq \underline{0} \qquad (1.7.4f)$$

$$p(\underline{u},\underline{u}) = 0, \text{ if and only if } \underline{u} = \underline{0} \qquad (1.7.4g)$$

---

* Note: conjugate linear in $\underline{v}$

Since

$$\mathrm{Im}\{\psi(\underline{u})\}\equiv\frac{1}{2}\left(\psi(\underline{u})-\bar{\psi}(\underline{u})\right)$$

then

$$\mathrm{Im}\{\psi(\underline{u})\}=0$$

On the other hand, if $\underline{A}$ is skew-Hermitian, then,

$$\psi(\underline{u})=\underline{u}^*\underline{A}^*\underline{u}=-\underline{u}^*\underline{A}\underline{u}$$

and

$$\bar{\psi}(\underline{u})=\underline{u}^*\underline{A}\underline{u}$$

Since

$$\mathrm{Re}\{\psi(\underline{u})\}\equiv\frac{1}{2}\left(\psi(\underline{u})+\bar{\psi}(\underline{u})\right)$$

then

$$\mathrm{Re}\{\psi(\underline{u})\}=0$$

thus proving the "if" part of the theorem.

Exercise 1.7.2 Prove the "only if" part of Theorem 1.7.2

What Theorem 1.7.2 states is very important, namely that Hermitian matrices are good candidates for defining a scalar product for a vector space, since the associated quadratic form is real. What is now left to investigate is whether this form turns out to be positive definite as well. Though this is not true for any Hermitian matrix, it is (obviouly!) so for positive definite Hermitian matrices (by definition!). Futhermore, since the quadratic form of a positive definite matrix must, in the first place, be real, and since, for the quadratic form associated with a matrix to be real, the matrix must be Hermitian (from Theorem 1.7.2), it is not necessary to refer to a positive definite (or semidefinite) matrix as being Hermitian.

Summarizing: In order for the quadratic form (1.7.2) to be a scalar product, $\underline{A}$ must be positive definite. Next, a very important result concerning an easy characterization of positive definite (semidefinite) matrices is given.

Substitution of (1.7.10) and (1.7.12) into (1.7.11) yields

$$\sum_{1}^{n}\lambda_i|u_i|^2 > (\geq) 0 \qquad (1.7.13)$$

Now, assume $\underline{u}$ is such that all but its $k^{\underline{th}}$ component vanish; in this case, (1.7.13) reduces to

$$\lambda_k|u_k|^2 > (\geq) 0$$

from which

$$\lambda_k > (\geq) 0$$

and, since λ_k can be any of the eigenvalues of $\underline{A}$, the proof of this part is done. The proof of the "if" part is obvious and is left as an exercise for the reader.

Exercise 1.7.2 Show that, if the eigenvalues of a square matrix are all real and greater than (or equal to) zero, the matrix is posite definite (semidefinite).

A very special case of a positive definite matrix is the identity matrix, $\underline{I}$, which yields the very well known scalar product

$$p(\underline{u},\underline{v}) = \underline{u}^*\underline{I}^*\underline{v} = \underline{u}^*\underline{I}\underline{v} = \underline{u}^*\underline{v} \qquad (1.7.14)$$

In dealing with vector spaces over the real field, the arising inner product is real and hence, from Schwarz's inequality (1.4, p.125).

$$\frac{p(\underline{u},\underline{v})}{\sqrt{p(\underline{u},\underline{u})p(\underline{v},\underline{v})}} \leq 1$$

thus making it possible to define a "geometry" for then, the cosine of the angle between vectors $\underline{u}$ and $\underline{v}$ can be defined as

$$\cos(\underline{u},\underline{v}) = \frac{p(\underline{u},\underline{v})}{\sqrt{p(\underline{u},\underline{u})p(\underline{v},\underline{v})}}$$

However, computing it requires n (the dimension of the space to which the vector under consideration belongs) multiplications (i.e. n square raisings), n-1 additions and one square root computation. In order to proceed further, some more definitions are needed.

An invertible linear transformation is called an "isometry" if it preserves the following scalar product

$$p(\underline{x},\underline{y}) = p(A\underline{x}, A\underline{y}) = \underline{x}^*\underline{A}^*\underline{A}\underline{y} \qquad (1.8.3)$$

It is a very simple matter to show that, in order for a transformation $\underline{P}$ to be an isometry, it is required that its transpose conjugate, $\underline{P}^*$, equals its inverse, i.e.,

$$\underline{P}^*=\underline{P}^{-1} \qquad (1.8.4)$$

If $\underline{P}$ is defined over the complex field and meets condition (1.8.4), then it is said to be unitary. If $\underline{P}$ is defined over the real field, then $\underline{P}^*=\underline{P}^T$, the transpose of $\underline{P}$ and, if it satisfies (1.8.4), it is said to be orthogonal.

Exercise 1.8.1 Show that in order for $\underline{P}$ to be an isometry, it is necessary that $\underline{P}$ satisfies (1.8.4), i.e., show that under the similarity transformation

$$\underline{\xi}=\underline{P}\underline{x},\ \underline{\eta} = \underline{P}\underline{y},\ \underline{B}=\underline{P}\underline{A}\underline{P}^{-1},$$

the following scalar product is preserved:

$$p(\underline{x},\underline{y}) = p(\underline{\xi},\underline{\eta})$$

1.9 PROPERTIES OF UNITARY AND ORTHOGONAL MATRICES.

Some important facts about unitary and orthogonal matrices are discussed in this section. Notice that all results concerning unitary matrices apply to orthogonal matrices, for the latter are a special case of the former.

THEOREM 1.9.1 *The set of eigenvalues of a unitary matrix lies on the unit circle $|z|^2 = 1$, centered at the origin of the complex plane.*

minimum, it is said to be a saddle point. Criteria to decide whether an extremum is a maximum, a minimum or a saddle point are next derived. An expansion of ϕ around $\underline{x}_0$ in a Taylor series illustrates the kind of stationary point at hand. In fact, the Taylor expansion of ϕ is

$$\phi(\underline{x}) = \phi(\underline{x}_0) + \phi'(\underline{x}_0)^T(\underline{x}-\underline{x}_0) + \frac{1}{2}(\underline{x}-\underline{x}_0)^T \phi''(\underline{x}_0)(\underline{x}-\underline{x}_0) + R \qquad (1.10.1)$$

where R is the residual, which contains terms of third and higher orders. Then the increment of ϕ at $\underline{x}_0$, for a given increment $\Delta\underline{x} = \underline{x}-\underline{x}_0$, is given by

$$\Delta\phi = \phi'(\underline{x}_0)^T\Delta\underline{x} + \frac{1}{2}\Delta\underline{x}^T\phi''(\underline{x}_0)\Delta\underline{x} \qquad (1.10.2)$$

if terms of third and higher orders are neglected.

From eq. (1.10.2) it can be concluded that the linear pant of $\Delta\phi$ vanishes at a stationary point, which makes clear why such points are called stationary Whether $\underline{x}_0$ constitutes an extremum or not, depends on the sign of $\Delta\phi$. It is a maximum if $\Delta\phi$ is nonpositive for arbitrary $\Delta\underline{x}$. It is a minimum if the said increment is nonnegative for arbitrary $\Delta\underline{x}$. If the sign of the increment depends on $\Delta\underline{x}$, then $\underline{x}_0$ is a saddle point for reasons which are brought up in the following. Eq. (1.10.2) shows that the sign of $\Delta\phi$ depends entirely on the quadratic term, at a stationary point. Whether this term is nonpositive or nonnegative, it is sufficient that the Hessian matrix $\phi''(x)$ be sign semidefinite at x_0. Notice, however, that this condition on the Hessian matrix is only sufficient, but not necessary, for it is based on Eq. (1.10.2), which is truncated after third-order terms. In fact, a function whose Hessian at a stationary point is sign-semidefinite can constitute either a maximum, a minimum, or a saddle point as shown next.

From the foregoing discussion, the following theorem is concluded.

THEOREM 1.10.1 *Extrema and saddle points of a differentiable function occur at stationary points. For a stationary point to constitute a local maximum (minimum) it is sufficient, although not necessary, that the*

Exercise 1.10.1 Prove Corollary 1.10.7

Example 1.10.1 The function $\phi = x_1^4 + x_2^4 + \ldots + x_n^4$ has a local minimum at $x_1 = x_2 = \ldots = x_n = 0$. The Hessian matrix of this function, however, vanishes at this minimum.

Example 1.10.2 The function $= x_1^4 - x_2^4$ has a stationary point at the origin, which is a saddle point. Its Hessian matrix, however, vanishes at this point.

Example 1.10.3 The function $x_1^2 + x_2^4$ has a minimum at $(0,0)$. At this point its Hessian matrix is positive semidefinite.

1.11 LINEAR ALGEBRAIC SYSTEMS.

Let $\underset{\sim}{A}$ be an mxn matrix and $\underset{\sim}{x}$ and $\underset{\sim}{b}$ be n-and m-dimensional vectors where, in general, $m \neq n$. Equation

$$\underset{\sim}{A}\underset{\sim}{x}=\underset{\sim}{b} \tag{1.11.1}$$

is a linear algebraic system. It is linear because, if $\underset{\sim}{x}_1$ and $\underset{\sim}{x}_2$ are its solutions for $\underset{\sim}{b}=\underset{\sim}{b}_1$ and $\underset{\sim}{b}=\underset{\sim}{b}_2$, and α and β are scalars, then $\alpha\underset{\sim}{x}_1+\beta\underset{\sim}{x}_2$ is a solution for $\underset{\sim}{b}=\alpha\underset{\sim}{b}_1+\beta\underset{\sim}{b}_2$. It is algebraic as opposed to differential or dynamic because it does not involve derivatives. There are three different cases regarding the solution of eq. (1.11.1), depending on whether m is greater than, less than or equal to n. These are discussed next:

i) $m>n$. In this case the number of equations is greater than that of unknowns. The system is overdetermined and there is no guarantee of the existence of a certain $\underset{\sim}{x}_0$ such that $\underset{\sim}{A}\underset{\sim}{x}_0=\underset{\sim}{b}$.

A very simple example of such a system is the following:

$$x_1=5 \tag{1.11.1a}$$

$$x_1=3 \tag{1.11.1b}$$

where $m=2$ and $n=1$. If $x_1=5$, the first equation is satisfied but the second one is not. If, on the other hand, $x_1=3$, the second equation is satisfied, but the first one is not. However, a system with $m>n$ could have a solution, which could even be unique if, out of the m

inverse" of $\underset{\sim}{A}$. A method to determine $\underset{\sim}{x}_0$ that does not require the computation of $\underset{\sim}{A}^I$ is given in (1.5) and (1.6). In (1.7), an iterative method to compute $\underset{\sim}{A}^I$ is proposed. The numerical solution of this problem is presented in section 1.12. This problem arises in such fieds as control theory, curve-fitting (regressions) and mechanism synthesis.

ii) m<n. In this case the number of equations is less than that of unknowns. Hence, if the system is consitent*, it has an infinity of solutions. For instance, the system

$$x+y=3, \tag{1.11.9}$$

in which m=1 and n=2, admits infinitely many solution, namely all points lying on the line

$$y=x+3 \tag{1.11.10}$$

Now consider the system

$$x+y+z=1 \tag{1.11.11a}$$

$$x+y-z=1 \tag{1.11.11b}$$

with m=2 and n=3. This system admits an infinity of solutions all with z=0.

In case a system with m<n admits a solution, it in fact admits infintely many, which is not difficult to prove. Indeed, partition matrix $\underset{\sim}{A}$ and vector $\underset{\sim}{x}$ in the form

$$\underset{\sim}{A}=\underbrace{\left[\underset{\sim}{A}_1 \vdots \underset{\sim}{A}_2\right]}_{m \quad n-m} \; m, \qquad \underset{\sim}{x}=\begin{matrix}\left[\begin{matrix}\underset{\sim}{x}_1 \\ \hdashline \underset{\sim}{x}_2\end{matrix}\right]\begin{matrix}\} m \\ \} n-m\end{matrix}\end{matrix}$$

Thus, eq. (1.11.1) is equivalent to

$$\underset{\sim}{A}_1\underset{\sim}{x}_1 + \underset{\sim}{A}_2\underset{\sim}{x}_2=\underset{\sim}{b} \tag{1.11.13}$$

---

* i.e. if $\underset{\sim}{b} \in R(\underset{\sim}{A})$

From which, if AA^T is of full rank,

$$\underline{\lambda}=-2(\underline{A}\underline{A}^T)^{-1}\underline{b} \qquad (1.11.21)$$

Finally, substituting the latter value of $\underline{\lambda}$ into eq. (1.11.19),

$$\underline{x}=\underline{A}^T\ (\underline{A}\underline{A}^T)^{-1}\underline{b}=\underline{A}^+\underline{b} \qquad (1.11.22)$$

where

$$\underline{A}^+\equiv\underline{A}^T\ (\underline{A}\underline{A}^T)^{-1}$$

is another pseudo-inverse of $\underline{A}$.

Exercise 1.11.1 Can both pseudo-inverses of $\underline{A}$, the one given in (1.11.8) and that of (1.11.23) exist for a given matrix $\underline{A}$? Explain.

The foregoing solution (1.11.22) has many interpretations: in control theory it yields the control taking a system from a known initial state to a desired final one while spending the minimum amount of energy. In Kinematics it finds two interpretations which will be given in Ch. 2, togehter with applications to hypoid gear design.

Exercise 1.11.2 Show that the image of the error (1.11.4) is perpendicular to $\underline{x}_0$ as given by (1.11.8). This result is known as the "Projection Theorem" and finds extensive applications in optimization theory.(1.9).

iii) m=n. This is the best known case and an extensive discussion of it can be found in any elementary linear algebra textbook. The most important result in this case states that if $\underline{A}$ is of full rank, i.e. if $\det \underline{A} \neq 0$, then the system has a unique solution, which is given by

$$\underline{x}=\underline{A}^{-1}\underline{b}$$

1.12 NUMERICAL SOLUTION OF LINEAR ALGEBRAIC SYSTEMS

Consider the system (1.11.1) for all three cases discussed in section 1.11.

i) m=n. The first case that will be discussed here is that for m=n.

There are many methods to solve such a linear algebraic system, but all

If $\underline{A}$ is not singular, the user calls the SOLVE subprogram, which computes the solution to the system by back substitution, i.e. from (1.12.1) in the following manner: The equation

$$\underline{L}\underline{U}\underline{x}=\underline{b} \qquad (1.12.2)$$

can be written as

$$\underline{L}\underline{y}=\underline{b}$$

by setting $\underline{U}\underline{x}=\underline{y}$. Thus

$$\underline{y}=\underline{L}^{-1}\underline{b}=\underline{c} \qquad (1.12.3)$$

where $\underline{L}^{-1}$ exists since det $\underline{L}$ (the product of the elements on the diagonal of $\underline{L}$) is equal to one (1.11). Substituting (1.12.3) into $\underline{U}\underline{x}=\underline{y}$, one obtains the final solution:

$$\underline{x}=\underline{U}^{-1}\underline{c}$$

where $\underline{U}^{-1}$ exists because $\underline{A}$ has been detected to be nonsingular*.

The flow diagram of the whole program appears in Fig 1.12.1 and the listings of DECOMP and SOLVE in Figs. 1.12.2 and 1.12.3

ii) m>n. Next, the numerical solution of the overdetermined linear system $\underline{A}\underline{x}=\underline{b}$ is discussed. In this case the number of equations is greater than that of unknowns and hence the sought "solution" is that $\underline{x}_0$ which minimizes the Euclidean norm of the error $\underline{A}\underline{x}_0-\underline{b}$. This is done by application of Householder reflections (1.5) to both $\underline{A}$ and $\underline{b}$. A Householder reflection is an orthogonal transformation $\underline{H}$ which has the property that

$$\underline{H}^{-1}=\underline{H}^{T}=\underline{H} \qquad (1.12.4)$$

Given an m-vector $\underline{a}$ with components $a_1, a_2, \ldots, a_m$, the Housholder reflection $\underline{H}$ (a function of $\underline{a}$) defined as

---

* In fact, there is no need to explicitely compute $\underline{L}^{-1}$ and $\underline{U}^{-1}$, for the triangular structure of $\underline{L}$ and $\underline{U}$ permits a recursive solution.

```
10          SUBROUTINE DECOMP(N,NDIM,A,IP)
11        REAL A(NDIM,NDIM),T
12        INTEGER IP(NDIM)
13 C
14 C     MATRIX TRIANGULARIZATION BY GAUSSIAN ELIMINATION
15 C
16 C      INPUT :
17 C      N      = ORDER OF MATRIX
18 C      NDIM   = DECLARED DIMENSION OF ARRAY A, IN THE MAIN PROGRAM
19 C      A      = MATRIX TO BE TRIANGULARIZED
20 C
21 C      OUTPUT :
22 C      A(I,J), I.LE.J  = UPPER TRIANGULAR FACTOR, U
23 C      A(I,J), I.GT.J  =MULTIPLIERS = LOWER TRIANGULAR FACTOR, I-L
24 C      IP(K), K.LT.N   =INDEX OF K-TH PIVOT ROW
25 C      IP(N)           = (-1)**(NUMBER OF INTERCHANGES) OR 0.
26 C      USE "SOLVE" TO OBTAIN SOLUTION OF LINEAR SISTEM
27 C      DETERM(A)       = IP(N)*A(1,1)*A(2,2)*...*A(N,N)
28 C      IF IP(N)=0, A IS SINGULAR, "SOLVE" WILL DIVIDE BY ZERO
29 C      INTERCHANGES FINISHED IN U, ONLY PARTLY IN L
30 C
31        IP(N)=1
32        DO 60 K=1,N
33           IF(K.EQ.N) GO TO 50
34           KP1=K+1
35           M=K
36           DO 10 I=KP1,N
37              IF(ABS(A(I,K)).GT.ABS(A(M,K))) M=I
38   10      CONTINUE
39           IP(K)=M
40           IF(M.NE.K) IP(N)=-IP(N)
41           T=A(M,K)
42           A(M,K)=A(K,K)
43           A(K,K)=T
44           IF(T.EQ.0) GO TO 50
45           DO 20 I=KP1,N
46   20         A(I,K)=-A(I,K)/T
47           DO 40 J=KP1,N
48              T=A(M,J)
49              A(M,J)=A(K,J)
50              A(K,J)=T
51              IF(T.EQ.0.) GO TO 40
52              DO 30 I=KP1,N
53   30            A(I,J)=A(I,J)+A(I,K)*T
54   40      CONTINUE
55   50      IF(A(K,K).EQ.0.) IP(N)=0
56   60   CONTINUE
57        RETURN
58        END
#
```

Fig. 1.12.2 Listing of SUBROUTINE DECOMP

$$\alpha = \text{sgn}(a_1)||\underset{\sim}{a}|| \qquad (1.12.5a)$$

$$\underset{\sim}{u} = \underset{\sim}{a}+\alpha\underset{\sim}{e}_1 \qquad (1.12.5b)$$

$$\beta = \alpha u_1 \qquad (1.12.5c)$$

$$\underset{\sim}{H} = \underset{\sim}{I} - \frac{1}{\beta}\underset{\sim}{u}\underset{\sim}{u}^T \qquad (1.12.5b)$$

Transforms $\underset{\sim}{a}$ into $-\alpha\underset{\sim}{e}_1$, and reflects any other vector $\underset{\sim}{b}$ about a hyperplane perpendicular to $\underset{\sim}{u}$.

On the other hand, if $\underset{\sim}{H}_k$ is defined as

$$\alpha_k = \text{sgn}(a_k)(a_k^2 + a_{k+1}^2+\ldots+a_m^2)^{\frac{1}{2}} \qquad (1.12.6a)$$

$$\underset{\sim}{u}_k = (0,\ldots,0,a_k+\alpha_k, a_{k+1},\ldots,a_m)^T \qquad (1.12.6b)$$

$$\beta_k = \alpha_k(\underset{\sim}{u}_k)_k \qquad (1.12.6c)$$

$$\underset{\sim}{H}_k = \underset{\sim}{I} - \frac{1}{\beta}\underset{\sim}{u}_k\underset{\sim}{u}_k^T \qquad (1.12.6d)$$

then $\underset{\sim}{H}_k\underset{\sim}{a}$ is a vector whose first k-1 components are identical to those of $\underset{\sim}{a}$, its k^{th} component is $-\alpha_k$ and its remaining m-k components are all zero. Furthermore, if v is any other vector, then

$$\underset{\sim}{H}_k\underset{\sim}{v} = \underset{\sim}{v} - \gamma\underset{\sim}{u}$$

where

$$\gamma = \frac{\underset{\sim}{v}^T\underset{\sim}{v}}{\beta}$$

and if, in particular, $v_k = v_{k+1} = \ldots = v_m = 0$, then

$$\underset{\sim}{H}_k\underset{\sim}{v} = \underset{\sim}{v}$$

Let now $\underset{\sim}{H}_i$ be the Householder reflection wich cancels the last m-i components of the i^{th} column of $\underset{\sim}{H}_{i-1}\underset{\sim}{A}$, while leaving its i-1 components unchanged and setting its i^{th} component equal to $-\alpha_i$, for i=1,...,n. By application of the n Householder reflections thus defined, on $\underset{\sim}{A}$ and $\underset{\sim}{b}$ in the form

$$\underset{\sim}{H}_n\underset{\sim}{H}_{n-1}\ldots\underset{\sim}{H}_2\underset{\sim}{H}_1\underset{\sim}{A}\underset{\sim}{x}_0 = \underset{\sim}{H}_n\underset{\sim}{H}_{n-1}\ldots\underset{\sim}{H}_2\underset{\sim}{H}_1\underset{\sim}{b} \qquad (1.12.7)$$

```
100           SUBROUTINE HECOMP(MDIM,M,N,A,U)
101           INTEGER MDIM,M,N
102           REAL A(MDIM,N),U(M)
103           REAL ALPHA,BETA,GAMMA,SQRT
104 C
105 C     HOUSEHOLDER REDUCTION OF RECTANGULAR MATRIX TO UPPER
106 C     TRIANGULAR FORM.  USE WITH HOLVE FOR LEAST-SQUARE
107 C     SOLUTIONS OF OVERDETERMINED SYSTEMS.
108 C
109 C     MDIM= DECLARED ROW DIMENSION OF A
110 C     M    = NUMBER OF ROWS OF A
111 C     N    = NUMBER OF COLUMNS OF A
112 C     A    = M-BY-N MATRIX WITH M.>.N
113 C            INPUT :
114 C                     MATRIX TO BE REDUCED
115 C            OUTPUT:
116 C                     REDUCED MATRIX AND INFORMATION ABOUT REDUCTION
117 C     U    = M-VECTOR
118 C            INPUT :
119 C                     IGNORED
120 C            OUTPUT:
121 C                     INFORMATION ABOUT REDUCTION
122 C
123 C           FIND REFLECTION WHICH ZEROES A(I,K), I= K+1,.........,M
124 C
125           DO 6 K= 1,N
126              ALPHA= 0.0
127              DO 1 I= K,M
128                 U(I)= A(I,K)
129                 ALPHA= ALPHA+U(I)*U(I)
130     1        CONTINUE
131              ALPHA= SQRT(ALPHA)
132              IF(U(K).LT.0.0) ALPHA= -ALPHA
133              U(K)= U(K)+ALPHA
134              BETA= ALPHA*U(K)
135              A(K,K)= -ALPHA
136              IF(BETA.EQ.0.0.OR.K.EQ.N) GO TO 6
137 C
138 C     APPLY REFLECTION TO REMAINING COLUMNS OF A
139              KP1= K+1
140              DO 4 J= KP1,N
141                 GAMMA= 0.0
142                 DO 2 I= K,M
143                    GAMMA= GAMMA+U(I)*A(I,J)
144     2           CONTINUE
145                 GAMMA= GAMMA/BETA
146                 DO 3 I= K,M
147                    A(I,J)= A(I,J)-GAMMA*U(I)
148     3           CONTINUE
149     4        CONTINUE
150     6     CONTINUE
151           RETURN
152 C
153 C     TRIANGULAR RESULT STORED IN A(I,J), I.LE.J
154 C     VECTORS DEFINING REFLECTIONS STORED IN U AND REST OF A
155           END
```

Fig 1.12.4 Listing of SUBROUTINE HECOMP

Exercise 1.12.3* Show that $\underset{\sim}{H}$, as defined in eqs. (1.12.5) is in fact a reflection, i.e. show that $\underset{\sim}{H}$ is orthogonal and the value of its determinant is -1. (Hint: Use the result of Exercise 1.12.2).

iii) m<n: Now, the linear system of equations $\underset{\sim}{A}\underset{\sim}{x}=\underset{\sim}{b}$ is studied when the number of unknowns is greater that the number of equations.

In this case, the system is underdetermined and has an infinity of solutions. However, as was discussed in Section 1.11, among those solutions, there is one, say $\underset{\sim}{x}_0$ whose Euclidean norm is a minimum. This is given by eq. (1.11.22), repeated here for ready reference.

$$\underset{\sim}{x}_0=\underset{\sim}{A}^T(\underset{\sim}{A}\underset{\sim}{A}^T)^{-1}\underset{\sim}{b} \tag{1.12.8}$$

One possible way of computing $\underset{\sim}{x}_0$ is given next:

a) Write eq. (1.11.20) in the form

$$\underset{\sim}{A}\underset{\sim}{A}^T\underset{\sim}{\lambda}=\underset{\sim}{b} \tag{1.12.9}$$

b) Using the LU decomposition method, find $\underset{\sim}{\lambda}$ from (1.12.9)

c) With $\underset{\sim}{\lambda}$ known from step ii), compute $\underset{\sim}{x}_0$ by matrix multiplication, as appearing in (1.11.19), i.e.

$$\underset{\sim}{x}_0=\underset{\sim}{A}^T\underset{\sim}{\lambda} \tag{1.12.10}$$

1.13 NUMERICAL SOLUTION OF NONLINEAR ALGEBRAIC SYSTEMS.

For several reasons, nonlinear systems are more difficult to deal with than are linear systems. Considering the simplest case of equal number of equations and unknowns, there is no guarantee that the nonlinear system has a uinque solution; in fact, there is no guarantee that the system has a solution at all.

---

* See Section 2.3 for more details on reflections.

Example 1.13.1 The 2nd order nonlinear algebraic system

$$x^2 - y^2 = 16 \qquad (a)$$

$$x^2 + y^2 = 1 \qquad (b)$$

has no solution, for the hyperbola (a) does not intersect the circle (b), as is shown in Fig. 1.13.1

Example 1.13.2 The 2nd order linear algebraic system

$$x^2 - y^2 = 1 \qquad (c)$$

$$x^2 + y^2 = 4 \qquad (d)$$

has four solutions, namely

$$x_1 = \sqrt{\frac{5}{2}},\ y_1 = \sqrt{\frac{3}{2}}$$

$$x_2 = \sqrt{\frac{5}{2}},\ y_2 = -\sqrt{\frac{3}{2}}$$

$$x_3 = -\sqrt{\frac{5}{2}},\ y_3 = -\sqrt{\frac{3}{2}}$$

$$x_4 = -\sqrt{\frac{5}{2}},\ y_4 = \sqrt{\frac{3}{2}}$$

which are the four points where the hyperbola (c) intersects the circle (d). These intersections appear in Fig. 1.13.2

The most popular method of solving a nonlinear algebraic system is the so-called Newton-Raphson method. First, the system of equations has to be written in the form

$$\underline{f}(\underline{x}) = \underline{0} \qquad (1.13.1)$$

where $\underline{f}$ and $\underline{x}$ are m- and n- dimensional vectors. For example, system (a), (b) of Example 1.13.1 can be written in the form

$$f_1(x_1, x_2) = x_1^2 - x_2^2 - 16 = 0 \qquad (a')$$

$$f_2(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0 \qquad (b')$$

Here f_1 and f_2 are the components of the 2-dimensional vectors $\underline{f}$ and x_1 and

which is the Newton-Raphson iterative scheme. The procedure is stopped when a convergence criterion is met. One possible criterion is that the norm of $\underset{\sim}{f}(\underset{\sim}{x}_k)$ reaches a value below certain prescribed tolerance, i.e.

$$||\underset{\sim}{f}(\underset{\sim}{x}_k)|| \leq \epsilon \qquad (1.13.8)$$

where ϵ is the said tolerance. On the other hand, it can also happen that at iteration k, the norm of the increment becomes smaller than the tolerance. In this case, even if the convergence criterion (1.13.8) is not met, it is useless to perform more interations. Thus, it is more reasonable to verify first that the norm of the correction does not become too small before proceeding further, and stop the procedure if both $||\underset{\sim}{f}(\underset{\sim}{x}_k)||$ and $||\Delta\underset{\sim}{x}_k||$ are small enough, in which case, convergence is reached.
If only $||\Delta\underset{\sim}{x}_k||$ goes below the imposed tolerance, do not accept the corresponding $\underset{\sim}{x}_k$ as the solution. The conditions under which the procedure converges are discussed in (1.15). These conditions, however, cannot be verified easily, in general. What is advised to do is to try different initial guesses $\underset{\sim}{x}_0$ till convergence is reached and to stop the procedure if either

i) too many iterations have been performed

or

ii) $||\Delta\underset{\sim}{x}_k|| \leq \epsilon$ but $||\underset{\sim}{f}(\underset{\sim}{x}_k)|| > \epsilon$

If the method of Newton-Raphson converges for a given problem, it does so quadratically, i.e. two digits are gained per iteration during the approximation to the solution. It can happen, however, that the procedure does not converge monotonically, in which case,

$$||\underset{\sim}{f}(\underset{\sim}{x}_{k-1})|| > ||\underset{\sim}{f}(\underset{\sim}{x}_k)||$$

thus giving rise to strong oscillations and, possibly, divergence. One way to cope with this situation is to introduce damping, i.e. instead of using

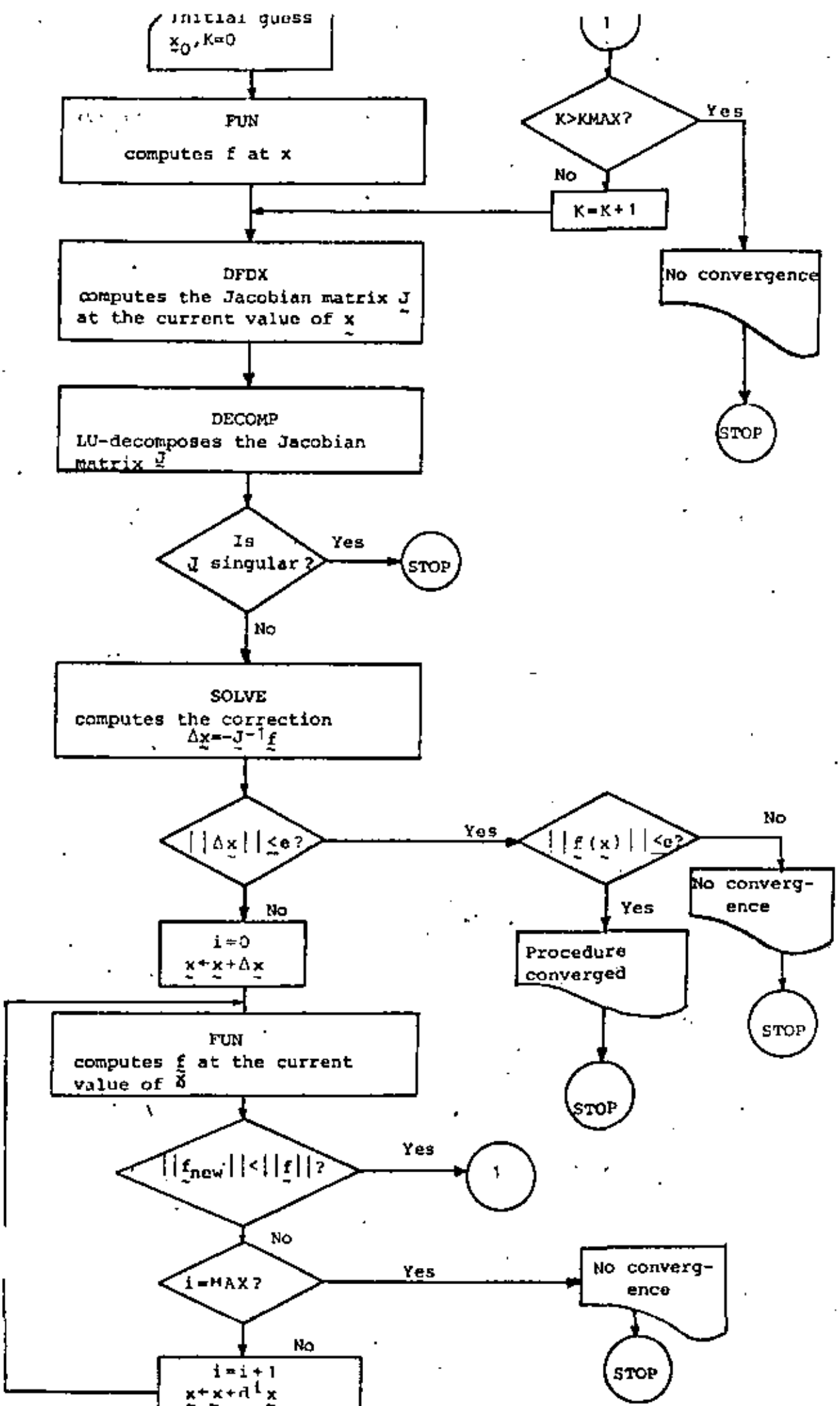


Fig. 1.13.3 Flow diagram to solve a nonlinear algebraic system with the same number of equations as of unknowns via the method of newton-Raphson with damping

```
1950          IF(DELNOR.LT.TOL) GO TO 8
1960          K=1
1970    3     DO 4 I=1,N
1980                X(I)=X(I)-DELTA(I)
1990    4     CONTINUE
2000          CALL FUN(X,F,P,N)
2010          FNOR2=FNORM(F,N)
2020  C
2030  C     TESTING THE NORM OF THE FUNCTION F AT CURRENT VALUE OF X. IF THIS
2040  C     DOES NOT DECREASE, THEN DAMPING IS INTRODUCED.
2050          IF(FNOR2.LT.TOL) GO TO 8
2060          IF(FNOR2.LT.FNOR1) GO TO 1
2070          IF(K.GT.KMAX) GO TO 7
2080          DO 6 I=1,N
2090               IF(K.GE.2) GO TO 5
2100               DELTA(I)=(DAMP-1.)*DELTA(I)
2110               GO TO 6
2120    5          DELTA(I)=DAMP*DELTA(I)
2130    6     CONTINUE
2140          K=K+1
2150          GO TO 3
2160    7     WRITE(6,101)
2170  C
2180  C     IT AT THIS ITERATION THE NORM OF THE FUNCTION CANNOT BE DECREASED
2190  C     AFTER KMAX DAMPINGS, DAMP IS SET EQUAL TO -1 AND THE SUBROUTINE
2200  C     RETURNS TO THE MAIN PROGRAM.
2210          DAMP=-1
2220          RETURN
2230    8     WRITE(6,102) FNOR2,ITER,K
2240          DO 9 I=1,N
2250               WRITE(6,103)I,X(I)
2260    9     CONTINUE
2270          RETURN
2280   10     WRITE(6,104)
2290          RETURN
2300   11     WRITE(6,105)
2310          DAMP=0
2320          RETURN
2330   101    FORMAT(10X,'NO CONVERGENCE WITH THIS DAMPING VALUE'/)
2340   102    FORMAT(2X,'CONVERGENCE REACHED. NORM OF THE FUNCTION',3X,
2350       -          6HFUN = F15.9//2X,'NUMBER OF ITERATIONS = ',I5,3X,
2360       -          'NUMBER OF DAMPINGS =',I3//5X,'THE SOLUTION IS:'/)
2370   103    FORMAT(5X,2HI=I6,10X,2HX=F10.4/)
2380   104    FORMAT(10X,'NO CONVERGENCE'/)
2390   105    FORMAT(10X,'JACOBI MATRIX IS SINGULAR'/)
2400          END
$
```

Fig 1.13.4 Listing of SUBROUTINE NRDAMP (Continued)

compute it from

$$\Delta \underset{\sim}{x}_k = -\alpha^i \left(\underset{\sim}{J}^T(\underset{\sim}{x}_k) \underset{\sim}{J}(\underset{\sim}{x}_k) \right)^{-1} \underset{\sim}{J}^T(\underset{\sim}{x}_k) \underset{\sim}{f}(\underset{\sim}{x}_k) \qquad (1.13.14)$$

for $i = 0, 1, \ldots,$ max and stop the damping when

$$||\phi(\underset{\sim}{x}_{k+1,i})|| < ||\phi(\underset{\sim}{x}_{k,i})||$$

The algorithm is illustrated with the flow diagram of Fig 1.13.5 and implemented with the subroutine NRDAMC, appearing in Fig 1.13.6

Third case: m<n

The system, in this case, is underdeterminated and infinitely many solutions can be expected to exist. Out of these solutions, however, one can choose that with a minimum norm, thus converting the problem into a nonlinear quadratic programming problem, stated as

$$\text{Minimize } \underset{\sim}{x}^T \underset{\sim}{x} \qquad (1.13.15a)$$

$$\text{subject to } \underset{\sim}{f}(\underset{\sim}{x}) = \underset{\sim}{0} \qquad (1.13.15b)$$

One way to find the minimizing $\underset{\sim}{x}_0$, of problem (1.13.15) is via the method of Lagrange multipliers. Thus, define a new objective function

$$\psi(\underset{\sim}{x}) = \underset{\sim}{x}^T \underset{\sim}{x} + \underset{\sim}{\lambda}^T \underset{\sim}{f}(\underset{\sim}{x}) \qquad (1.13.16)$$

which is stationary at $\underset{\sim}{x}_0$ where its gradient vanishes. Thus,

$$\psi(\underset{\sim}{x}_0) = 2\underset{\sim}{x}_0 + \underset{\sim}{f}'^T(\underset{\sim}{x}_0)\underset{\sim}{\lambda} = \underset{\sim}{0} \qquad (1.13.17)$$

The systems of equations (1.13.15b) and (1.13.17) now represent a larger system of m+n equations (m in (1.13.10b) and n in (1.13.12)) in m+n unknowns (m components of $\underset{\sim}{\lambda}$ and n components of $\underset{\sim}{x}$). Hence, the problem now reduces to the first case and so can be solved by application of the subroutine NRDAMP.

Exercise 1.13.2 Let

$$f(\underset{\sim}{x}) = \frac{1}{2} x_1^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + (x_3 - x_4)^2 +$$
$$+ \beta\left(\frac{7}{2}\cos x_1 + \frac{5}{2}\cos x_2 + \frac{3}{2}\cos x_3 + \frac{1}{2}\cos x_4\right)$$

be a scalar function of a vector argument $\underset{\sim}{x} = (x_1, x_2, x_3, x_4)^T$. Find its

```
      SUBROUTINE NRDAMC(X,FUN,DFDX,P,TOL,DAMP,N,M,ITER,MAX,KMAX)
      REAL X(N),F(M),DF(M,N),P(1),U(M),DELTA(M),FNORM1,FNORM2,
     -     DELNOR
C
C   THIS PROGRAM OBTAINS THE LEAST SQUARE SOLUTION TO THE NONLINEAR
C   SYSTEM F(X)= 0, WHERE F AND X ARE M-AND N-DIMENSIONAL VECTORS, M
C   BEING GREATER THAN N. THE PROCEDURE IS ITERATIVE, AND AT EACH
C   ITERATION FINDS THE LEAST SQUARE SOLUTION TO THE LINEAR SYSTEM
C   DF*DELTA= -F, WHERE DF IS THE JACOBIAN M X N MATRIX OF THE ORIGINAL
C   SYSTEM, COMPUTED AT THE CURRENT VALUE OF X. THE LINEAR LEAST SQUARE
C   SOLUTION AT EACH ITERATION IS FOUND VIA HOUSEHOLDER REFLECTIONS (
C   BJOERCK A. AND G. DAHLQUIST, NUMERICAL METHODS, PRENTICE HALL,
C   ENGLEWOOD CLIFFS, N.J., 1974, PP. 201-206, 443-444).
C   PARAMETERS :
C      X     ..= N-DIMENSIONAL VECTOR OF UNKNOWNS.
C      F     ..= M-DIMENSIONAL VECTOR OF FUNCTIONS WHOSE EUCLIDEAN NORM
C                IS TO BE MINIMIZED.
C      DF    ..= M X N JACOBIAN MATRIX OF F WITH RESPECT TO X.
C      P     ..= VECTOR OF PARAMETERS APPEARING IN FUN AND/OR DFDX. ITS
C                DIMENSION VARIES FROM PROBLEM TO PROBLEM.
C      TOL   ..= A REAL POSITIVE 'SMALL' VARIABLE DENOTING THE IMPOSED
C                TOLERANCE. IT IS SUPPLIED BY THE USER.
C      DAMP  ..= A REAL POSITIVE VARIABLE, 0.LT. DAMP .LT.1. IT DENOTES
C                THE DAMPING FACTOR AND IS SUPPLIED BY THE USER.
C      ITER  ..= AN INTEGER VARIABLE DENOTING THE NUMBER OF THE CURRENT
C                ITERATION.
C      MAX   ..= AN INTEGER VARIABLE DENOTING THE MAXIMUM NUMBER OF
C                ALLOWED ITERATIONS.
C      KMAX  ..= AN INTEGER VARIABLE DENOTING THE MAXIMUM NUMBER OF
C                ALLOWED DAMPINGS.
C   SUBSIDIARY SUBROUTINES :
C
C   HECOMP = TRIANGULARIZES A RECTANGULAR MATRIX BY HOUSEHOLDER
C            REFLECTIONS (MOLER C. B., MATRIX EIGENVALUE AND LEAST-
C            SQUARE COMPUTATIONS, COMPUTER SCIENCE DEPARTAMENT,
C            STANFORD UNIVERSITY. MARCH, 1973.)
C   HOLVE  = SOLVES TRIANGULARIZED SYSTEM BY BACK-SUBSTITUTION (MOLER
C            C. B., OP. CIT.)
C   FUN    = COMPUTES F.
C   DFDX   = COMPUTES DF.
C   FNORM  = COMPUTES THE MAXIMUM NORM OF A VECTOR.
C
C
      ITER=0
      CALL FUN(X,F,P,M,N)
    1 ITER=ITER+1
      IF(ITER.GT.MAX) GO TO 10
C
C   FORMS LINEAR LEAST SQUARE PROBLEM
      FNORM1=FNORM(F,M)
      CALL DFDX(X,DF,P,M,N)
      CALL HECOMP(M,M,N,DF,U)
      CALL HOLVE(M,M,N,DF,U,F)
```

Fig 1.13.6 Listing of SUBROUTINE NRDAMC

stationary points and decide whether each is either a maximum, a minimum or a saddle point, for $\beta = 1,10,50$.

Note: $f(x)$ could represent the potential energy of a mechanical system. In this case the stationary points correspond to the following equilibrium states: minima yield a stable equilibrium state, whereas maxima and saddle points yield unstable states.

Example 1.13.3 Find the point closest to all three curves of Fig 1.13.7. These curves are the parabola(P), the circle(C) and the hyperbola(H) with the following equations:

$$y = \frac{1.}{2.4} x^2 \qquad \text{(P)}$$

$$x^2 + y^2 = 4 \qquad \text{(C)}$$

$$x^2 - y^2 = 1 \qquad \text{(H)}$$

From Fig 1.13.7 it is clear that no single pair (x,y) satisfies all three equations simultaneously. There exist points of coordinates x_0, y_0, however, that minimize the quadratic norm of the error of the said equations. These can be found with the aid of SUBROUTINE NRDAMC. A program was written that calls NRDAMC, HECOMP and HOLVE to find the least-square solution to eqs. (P), (C) and (H). The found solutions were:

First solution: x=-1.61537, y=1.17844

Second solution: x= 1.61537, y=1.17844

which are shown in Fig 1.13.7. These points have symmetrical locations, as expected, and lie almost on the circle at abount equal distances from A_i and C_i and B_i and D_i (i=1,2)

The maximum error of the foregoing approximation was computed as 0.22070

R E F E R E N C E S

1.1 Lang S., Linear Algebra, Addison-Wesley Publishing Co., Menlo Park, 1970, pp. 39 and 40.

1.2 Lang S., op. cit., pp. 99 and 100

1.3 Finkbeiner, D.F., Matrices and Linear Transformations, W.H. Freeman and Company, San Francisco, 1960, pp. 139-142

1.4 Halmos, P.R., Finite-Dimensional Vector Spaces, Springer-Verlag, N. York, 1974.

1.5 Businger P. and G.H. Golub, "Linear Least Squares Solutions by Householder Transformations", in Wilkinson J.H. and C. Reinsch, eds., Handbook for Automatic Computation, Vol. II, Springer-Verlag, N. York, 1971, pp. 111-118

1.6 Stewart, G.W., Introduction to Matrix Computations, Academic Press, N.York, 1973, pp. 208-249.

1.7 Soderstrom T. and G.W. Stewart, "On the numerical properties of an iterative method for computing the Moore-Penrose generalized inverse", SIAM J. on Numerical Analysis, Vol. II, No. 1, March 1974.

1.8 Brand L., Advanced Calculus, John Wiley and Sons, Inc., N. York, 1955, pp. 147-197.

1.9 Luenberger, D.G., Optimization by Vector Space Methods, John Wiley and Sons, Inc., N. York, 1969, pp. 8, 49-52

1.10 Varga, R.S., Matrix Iterative Analysis, Prentice Hall, Inc., Englewood Cliffs, 1962, pp. 56-160

1.11 Forsythe, G.E. and C.B. Moler, Computer Solution of Linear Algebraic Systems, Prentice Hall, Inc., Englewood Cliffs, 1967, pp. 27-33

1.12 Moler C.B., "Algorithm 423. Linear Equation Solver (F 4)" Communications of the ACM, Vol. 15, Number 4, April 1973, p. 274.

1.13 Björck Å. and G. Dahlquist, Numerical Methods, Prentice-Hall, Inc., Englewood Cliffs, 1974, pp. 201-206.

1.14 Moler C.B., Matrix Eigenvalue and Least Square Computations, Computer Science Department, Stanford University, Stanford, California, 1973 pp. 4.1-4.15

1.15 Isaacson, E. and H. B. Keller, Analysis of Numerical Methods, John Wiley and Sons, Inc., N. York, 1966, pp. 85-123

1.16 Angeles, J., "Optimal synthesis of linkages using Householder reflections", Proceedings of the Fifth World Congress on the Theory of Machines and Mechanisms, vol. I, Montreal, Canada, July 8-13, 1979, pp. 111-114.

OPTIMACION DE SISTEMAS FISICOS

CONDICIONES DE OPTIMALIDAD

M. EN C. SUSANA GOMEZ GOMEZ

MARZO, 1981

C O N D I C I O N E S D E O P T I M A L I D A D

I. EL PROBLEMA GENERAL DE PROGRAMACION NO LINEAL

En el sentido más amplio, el problema general no lineal es el de encontrar un extremo (máximo o mínimo) de una función objetivo sujeta a restricciones de igualdad $^{y}/_{o}$ no lineales. Sin embargo, en las siguientes secciones de estas notas han quedado excluidos los dos siguientes problemas:

a) Las variables estan restringidas a valores enteros (programación no lineal entera)

b) Las restricciones incluyen al parámetro tiempo en la forma de una ecuación diferencial (control óptimo).

En lo siguiente se supondrá que la función objetivo $f(x)$ es continua, $h_1(x), \ldots, hm(x)$ denotan las restricciones de igualdad y $g_{m+1}(x), \ldots, g_p(x)$ las restricciones de desigualdad, donde: $x = (x_1, \ldots, x_n)^T$ es un vector columna de componentes $x_1, \ldots, x_n$ en un espacio euclidiano n-dimensional. (Las variables $x_1, x_2, \ldots, x_n$ pueden ser parámetros de diseño, ajuste de controles, lecturas de

instrumentos, etc., mientras que la función objetivo podría representar el costo, peso, ganancias, etc.; finalmente, las restricciones pueden representar requerimientos técnicos, condiciones de operación, etc., del proceso).

El problema de programación no lineal se puede establecer formalmente como:

$$\text{Minimizar:} \quad f(x), \; x \in E^n \tag{1.1}$$

sujeta a m restricciones de igualdad, lineales y/o no lineales,

$$h_j(x) = 0 \qquad j = 1, \ldots, m \tag{1.2}$$

y $(p - m)$ restricciones de desigualdad, lineales y/o no lineales,

$$g_j(x) \geq 0 \qquad j = m+1, \ldots, p \tag{1.3}$$

O en forma alterna como:

$$\text{Minimizar:} \quad \{f(x) \mid x \in R\} \tag{1.4}$$

donde R es el dominio de x para el cual (1.2) y (1.3) se satisfacen, es decir:

$$R = \{x \mid h_j(x) = 0, \; g_j(x) \geq 0, \text{ para toda } j\} \tag{1.5}$$

Un ejemplo sencillo de programación no lineal es el que se muestra en la figura 1, y está dado por:

Minimice: $f(x) = x_1^2 + x_2^2 + 2 \times 2$

sujeto a: $h_1(x) = x_1^2 + x_2^2 - 1 = 0$

$g_2 = x_1 + 2x2 - \frac{1}{2} \geq 0$

$g_3 = x_1 \geq 0$

$g_4 = x_2 \geq 0$

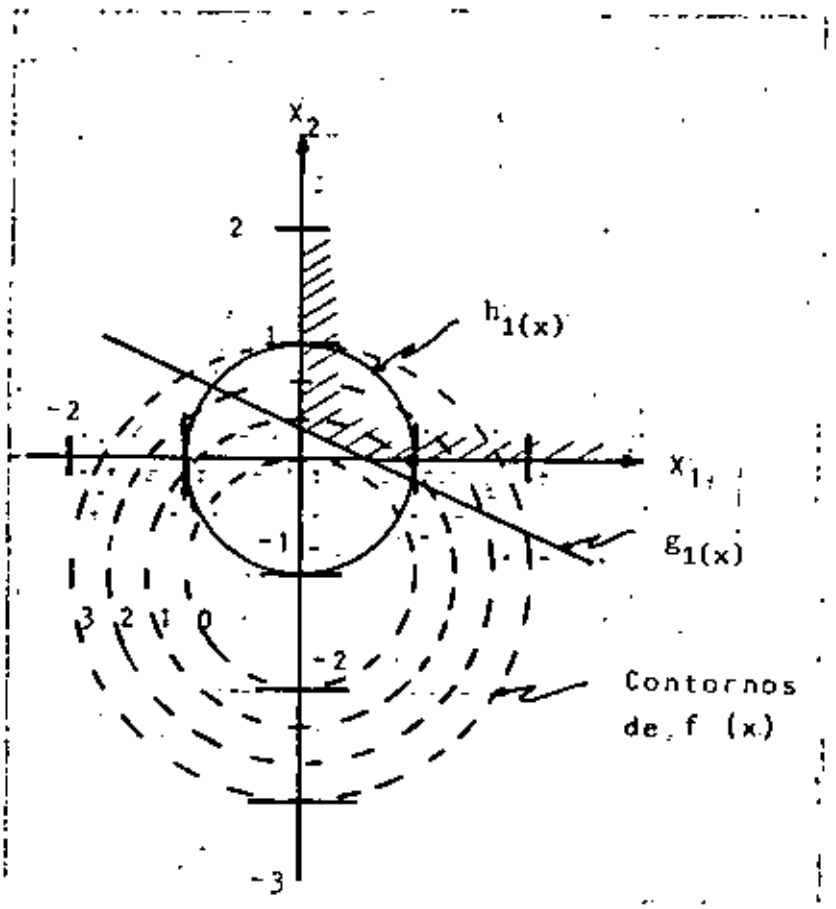


FIGURA 1. Representación geométrica de un problema de programación

II. NOTACION Y TERMINOLOGIA

El vector columna $x^* = (x_1^*, \ldots, x_n^*)^T$ que satisface (1.1) - (1.3) se denomina punto óptimo, y el valor de $f(x^*)$ que le corresponde se denomina valor óptimo de la función objetivo. La pareja, x^*, $f(x^*)$ constituye la solución óptima. Para algunos problemas, pueden existir varias categorias de soluciones óptimas si la función objetivo no es unimodal (exhibe mas de un punto extremo) como se ilustra en la figura 2. La solución global óptima representa el valor más pequeño de $f(x)$, mientras que una solución óptima local (o relativa) representa el valor más pequeño de $f(x)$ en una cierta vecindad del vector x: es decir,

óptimo global x^* satisface $\quad f(x^*) < f(x) \; \forall \, x \in E^n$

óptimo local x^* satisface $\quad f(x^*) < f(x) \; ||x-x^*|| \leq \sigma(x^*)$

2.1. Concavidad y Convexidad

Los conceptos de concavidad y convexidad ayudan a determinar bajo que condiciones una solución óptima local es tambien solución óptima global.

Una función $\phi(x)$ se dice que es convexa en el dominio R, si para cualesquiera dos vectores x_1 y x_2 $\in R$,

$$\phi(\theta x_1 + (1-\theta)x_2) \leq \theta\phi(x_1) + \phi(x_2)(1-\theta) \qquad (1.6)$$

donde θ es un escalar $0 \leq \theta \leq 1$. Además, $\phi(x)$ es estrictamente convexa si, para $x_1 \neq x_2$, el signo $\leq$ en (1.6) se puede remplazar por el signo de desigualdad $(<)$. Si en (1.6) la desigualdad contraria es la válida, se dice que la función $\phi(x)$ es cóncava $(\geq)$ o estrictamente cóncava $(>)$. Note que si $\phi(x)$ es cóncava (cónvexa), $-\phi(x)$ es cónvexa (cóncava). (Las funciones lineales son simultáneamente, convexas y cóncavas).

Una función convexa diferenciable posee las siguientes propiedades.

$$\left.\begin{array}{l}
\text{a) } \phi(x_2) - \phi(x_1) \geq \nabla^T\phi(x_1)(x_2-x_1) \quad \text{para toda } x_1 \text{ y } x_2. \\
\text{b) La matriz de segundas derivadas parciales de } \phi(x) \text{ con respecto a } x \text{ (matriz Hessiana) es positiva definida (o positiva semidefinida) para toda } x \text{ si } \phi(x) \text{ es estrictamente convexa (o convexa).} \\
\text{c) Sobre el dominio de } R, \ \phi(x) \text{ posee un sólo mínimo.}
\end{array}\right\} \quad (1.7)$$

Un conjunto de puntos (o región) se define como conjunto convexo en un espacio n-dimensional si, para toda pareja de puntos x_1 y x_2 en el conjunto, la línea recta que los une pertenece completamente al conjunto. Es decir, R es convexo si para toda x_1 y $x_2 \in R$

$$x = \theta x_1 + (1-\theta)x_2 \in R$$

De los conceptos de convexidad emerge un resultado importante en programación matemática: para el problema de programación no lineal conocido como el problema de programación convexa

$$\text{Minimizar} : \quad f(x)$$

$$\text{sujeta a} : \quad g_j(x) \geq 0 \qquad j = 1, \ldots, p$$

$$x \geq 0$$

en el cual (1), $f(x)$ es una función convexa y (2) cada restricción de desigualdad es una función cóncava (las restricciones forman un conjunto convexo), se puede demostrar el siguiente resultado: el mínimo local también es mínimo global (Usando argumentos opuestos, el resultado opuesto, máximo, también es cierto).

2.2. Factibilidad

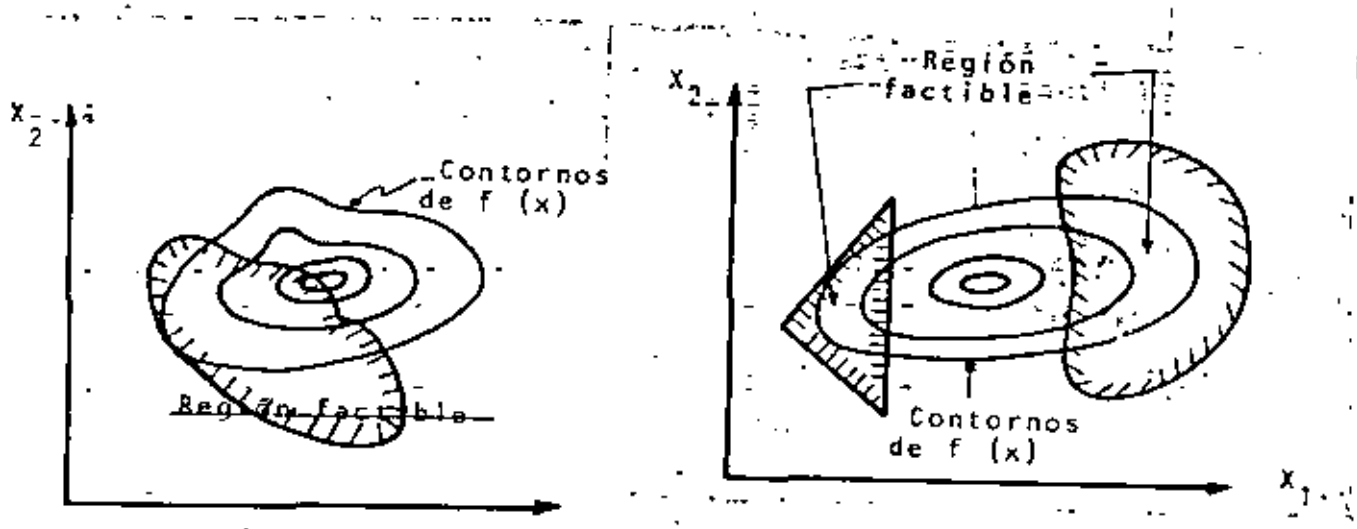
Cualquier vector x que satisface tanto las restricciones de desigualdad como las de igualdad se llama punto factible. El conjunto de todos los puntos que satisfacen las restricciones constituyen el dominio factible de $f(x)$, y se denotará por R: cualquier punto no en R se llama punto no factible.

Un óptimo restringido es uno para el cual el óptimo local cae en la frontera de la región factible. Si las restricciones son unicamente

de igualdad, un punto x factible debe caer en la intersección de todas las hipersuperficies que satisfacen $h_j(x) = 0$

Con respecto a las restricciones de desigualdad, un punto x se puede clasificar como <u>punto interior</u> (factible), <u>punto frontera</u> (factible) o <u>punto exterior</u> (no factible). Los puntos interiores son aquellos para los cuales $g_j(x)$ 0 ; para un punto frontera, g_j $g_j(x) = 0$ para al menos una restricción ; y un punto exterior, $g_j(x) > 0$ para al menos una restricción. Las restricciones se llaman <u>activas</u> (o de atadura) si $g_j(x) = 0$.

Una región R de vectores admisibles puede ser <u>convexa</u> o <u>no convexa</u>, según se describió con anterioridad, pero además puede ser <u>simplemente conexa</u> o <u>no-simplemente conexa</u> . (ver Figura 2).



(a) simplemente conexa (no convexa)

(b) no simplemente conexa (<u>obviamente</u> no convexa)

FIGURA 2. Ejemplos de tipos de región

2.3. El Gradiente

El conjunto de puntos para los cuales una función $f(x)$ exhibe un valor constante, se llaman contornos de $f(x)$. Si la función $f(x)$ es continua y diferenciable, el gradiente de la función existe y está definido como el vector columna formado por las primeras derivadas parciales de $f(x)$ con respecto a x es decir:

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} \qquad (1.8)$$

Se puede demostrar que en el espacio métrico euclidiano, el gradiente de una función escalar apunta en la dirección de máximo incremento en el valor de la función, máximo ascenso, y que es, además, ortogonal a las líneas de contorno. El negativo del gradiente apunta en la dirección del máximo descenso de $f(x)$. Finalmente, cualquier vector V, ortogonal a $\nabla f(x)$, tal como la superficie tangente a $f(x)$, está definido por

$V^T \nabla f(x) = 0$ (Figura 3)

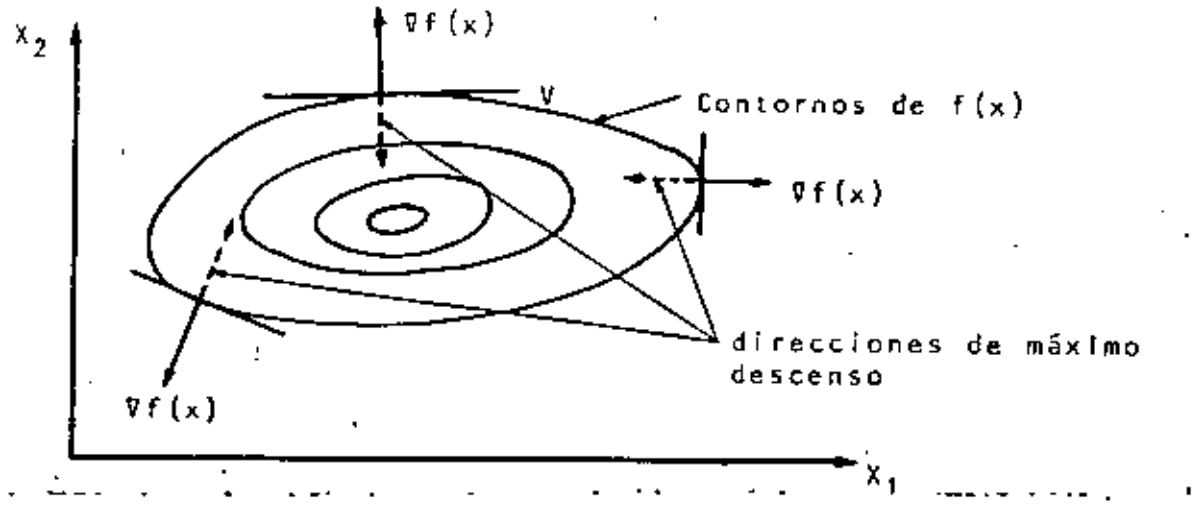


FIGURA 3. El gradiente, y la dirección de máximo descenso.

2.4. Aproximación de funciones

Algunos de los procedimientos de programación matemática que se discutirán más tarde requieren de aproximaciones lineales o cuadráticas para $f(x)$, $g_i(x)$, y $h_i(x)$.

Una aproximación lineal, o de primer orden, para una función $f(x)$, se puede hacer truncando la serie de Taylor, alrededor de un punto x_0, como

$$f(x) = f(x_0) + \nabla^T f(x_0)(x-x_0) \qquad (1.9)$$

Para obtener una aproximación cuadrática, en la misma serie de Taylor se pueden despreciar los términos mayores e iguales a tercer orden, obteniéndose

$$f(x) = f(x_0) + \nabla^T f(x_0)(x-x_0) + \frac{1}{2}(x-x_0)^T \nabla^2 f(x_0)(x-x_0) \qquad (1.10)$$

donde $\nabla^2 f(x)$ es la matriz Hessiana a $f(x)$, $H(x)$, es decir

$$H(x_0) = \{h_{ij}(x_0)\} \quad i,j = 1, \ldots, n \qquad (1.11a)$$

donde

$$h_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j} \qquad (1.11b)$$

Se observa facilmente que $H(x)$ es una matriz simétrica.

2.5. Condiciones Necesarias y Suficientes para que Una Solución sea Solución Optima

Para algunas clases especiales del problema general de programación no lineal (Ecs (1.1) - (1.3)) ha sido posible establecer criterios de optimalidad. Sin embargo, para funciones completamente generales, no ha sido posible establecer criterios de optimalidad precisos. En consecuencia, unicamente se describirán algunos casos especiales, los cuales, sin embargo, son bastante comunes y de importancia práctica. Las condiciones que determinan si un vector x resuelve o no el problema de programación no lineal serán presentadas en una serie de teoremas los cuales no serán demostrados (las demostraciones están fuera de los objetivos de este curso).

2.5.1. Programación no-lineal sin restricciones

El problema es el siguiente:

$$\text{Minimizar:} \quad f(x),\ x \in E^n \tag{1.12}$$

Las condiciones necesarias para que x^* sea un mínimo local del problema (1.12) son:

1. $f(x)$ diferenciable en x^*.
2. $\nabla f(x^*) = 0$, es decir, existe un punto estacionario de $f(x)$ en x^*

Las condiciones suficientes para que x^* sea un mínimo local del problema (1.12) son:

3. $\nabla^2 f(\vec{x}) > 0$; es decir, la matriz Hessiana es positiva definida.

(Las condiciones para la existencia de un máximo son iguales, excepto que el hessiano deberá ser negativo definido).

2.5.2. Programación no-lineal con restricciones de Igualdad y Desigualdad

En este caso el problema es el siguiente:

$$\begin{array}{lllll} \text{Minimizar:} & f(x) & & x \quad E^n & \\ \text{Sujeta a :} & h_j(x) = 0 & & j = 1, \ldots, m & \\ & g_j(x) \quad 0 & & j = m+1, \ldots, p & \end{array} \qquad (1.13)$$

Las condiciones necesarias para que x^* sea un mínimo local se establecen en dos teoremas, el primero de los cuales (teorema 2) puede ser llamado condiciones de primer orden (debido a que las funciones que intervienen se consideran una vez diferenciables). El segundo teorema (teorema 3) se le denomina condiciones de segundo orden (se considera que las funciones son dos veces diferenciables).

Para establecer las condiciones necesarias, empezaremos con el siguiente concepto: si x^* es un mínimo local de $f(x^*)$, ésta no puede decrecer a lo largo de ningun arco "suave" dirigido desde x^* hacia la región factible. Sea el vector V tangente al arco que empieza en x^*. Usando los conceptos de Fiacco y Mc Cormick, se asignan tres diferentes categorias o clases al vector V. donde cada conjunto V_i incluye el conjunto de V tales que: (ver tabla I).

Todas las posibles perturbaciones de x^* caen en la unión de V_1 y V_2 y si $V \in V_2$, $f(x)$ decrece, mientras que si $V \in V_1$, $f(x)$ se incrementa o es constante. En esencia, las condiciones necesarias de primer orden imponen el requerimiento de que el conjunto V_2 esté vacio.

	RESTRICCIONES			
CLASE	DESIGUALDAD		IGUALDAD	FUNCION OBJETIVO
V_1	$(\ v^T \nabla g_j(x^*) \geq 0$ para las restricciones activas$\ )$	y	$(\ v^T \nabla h_j(x^*) = 0,\ j = 1,2,\ldots,m\ )$	y $(\ v^T \nabla f(x^*) \geq 0\ )$
V_2	$(\ v^T \nabla g_j(x^*) \geq 0$ para las restricciones activas$\ )$	y	$(\ v^T \nabla h_j(x^*) = 0,\ j = 1,2,\ldots,m\ )$	y $(\ v^T \nabla f(x^*) < 0\ )$
V_3	$(\ v^T \nabla g_j(x^*) \leq 0$ para, al menos, una restricción activa$\ )$	o	$(\ v^T \nabla h_j(x^*) \neq 0$ para, al menos, una restricción$\ )$	

TABLA I. Clasificación de los conjuntos V_i

Si V_2 está vacio, se puede demostrar la existencia de los multiplicadores de Lagrange, resultando el siguiente teorema.

TEOREMA I.

Si (a) x^* satisface el problema (1.13), (b) las funciones $f(x)$, $g_j(x)$ son una vez diferenciables, y (c) en x^* V_2 está vacio, entonces existen los vectores u^* y w^* (multiplicadores de Lagrange) tales que, (u^*, w^*, x^*) satisfacen

$$(1) \quad h_j(x^*) = 0 \qquad j = 1, \ldots, m$$

$$(2) \quad g_j(x^*) \geq 0 \qquad j = m+1, \ldots, p$$

$$(3) \quad u_j^* g_j(x) = 0 \qquad j = m+1, \ldots, p$$

$$(4) \quad u_j^* \geq 0 \qquad j = m+1, \ldots, p$$

$$(5) \quad \nabla L(x^*, u^*, w^*) = 0$$

donde la función

$$L(x, u, w) = f(x) + \sum_{j-1}^{m} w_j h_j(x) - \sum_{j=m+1}^{p} u_j g_j(x)$$

puede ser considerada como una función Lagrangiana generalizada, asociada al problema (1.13).

Con el propósito de establecer bajo que circunstancias el conjunto V_2 está vacio, se necesita calificar, a primer orden, a las restricciones.

Sea x^* un punto factible del problema (1.13) y supóngase que $h_1(x), \ldots, hm(x), g_{m+1}(x), \ldots, g_p(x)$ son funciones una vez diferenciables. La calificación a primer orden de las restricciones es una condición que se impone unicamente sobre las restricciones (sin importar la función objetivo) y que consiste en que para cada punto frontera formado por el conjunto de restricciones de igualdad y las activas de desigualdad, debe de existir una curva suave que termine en el punto frontera y que pertenezca completamente al conjunto de las restricciones. Si x^* es un mínimo local de $f(x)$, ésta no puede decrecer a lo largo de tal curva, dirigida desde x^* hacia la región factible. Una condición suficiente para la calificación a primer orden de las restricciones que debe cumplirse es que todos los gradientes de las restricciones de desigualdad activas y los gradientes de las restricciones de igualdad evaluados en x^*, sean linealmente independientes. Esto último se establece en el siguiente teorema:

TEOREMA 2.

Si las funciones $h_1(x), \ldots, hm(x), g_{m+1}(x), \ldots, g_p(x)$ son una vez diferenciables en x^*, y si la calificación a primer orden de las restricciones es válida en x^*, entonces la condición necesaria para que x^* sea un mínimo local del problema (1.7), es que existan multiplicadores de Lagrange u^* y w^* tales que (u^*, w^*, x^*) satisfagan las ecuaciones (1) - (5) del Teorema 1.

Para tomar en cuenta la curvatura de las funciones en el problema (1.13), Mc Cormick estableció las condiciones necesarias de segundo orden para que x^* sea un mínimo local. Supongase que las funciones $f(x)$, $h_1(x)$, ..., $hm(x)$, $g_{m+1}(x)$, ..., $g_p(x)$ son dos veces diferenciables en x^*, un punto que satisface al problema (1.13). Sea V cualquier vector no cero tal que

$V^T \nabla g_j(x) = 0$ para las restricciones de desigualdad activas

$V^T \Delta h_j(x) = 0.$ para las restricciones de igualdad

Entonces, si V es la tangente a una curva $\psi(0)$, $\theta \geq 0$, dos veces diferenciable, a lo largo de la cual $g_i\{\psi(0)\} = 0$ para todas las restricciones de desigualdad activas, y $h_j\{\psi(0)\} = 0$ para todas las restricciones de igualdad, la calificación a segundo orden de las restricciones en x^* es válida. Una <u>condición suficiente para la calificación a segundo orden de las restricciones</u> que debe cumplirse es que los gradientes de las restricciones de desigualdad activas en x^* y los gradientes de las restricciones de igualdad en x^* sean linealmente independientes.

Las <u>condiciones necesarias de segundo orden</u> se pueden establecer como sigue.

TEOREMA 3.

(a) Si las funciones $f(x)$, $h_1(x)$, ..., $hm(x)$, $g_{m+1}(x)$, ..., $g_p(x)$ son dos veces diferenciables en x^*, y (b) si la calificación a primer orden de las restricciones es válida en x^*, entonces las condiciones necesarias para que x^* sea un mínimo local del problema (1.13) son que existan u^* y w^* tales que (c) ecuaciones (1) - (5) del Teorema 1 se satisfagan, y (d) para cada vector no cero V, para el cual $V^T g_j(x^*) = 0$, para las restricciones de desigualdad activas, y $V^T h_j(x^*) = 0$, para las restricciones de igualdad, se cumple lo siguiente:

$$(6) \quad V^T \nabla L \quad (x^*, u^*, w^*) \quad V \geq 0$$

Las condiciones suficientes para que x^* sea un mínimo local aislado del problema (1.13) son las mismas (a), (b), y (c) del Teorema 3, excepto la parte (d) (ecuación (6)), la cual debe ser substituida por:

(d'). Para cada vector V no cero para el cual $V^T \nabla g_j(x^*) = 0$ para las restricciones de desigualdad activas, $V^T \nabla g_j(x^*) \geq 0$ para las restricciones de desigualdad no activas y $V^T \nabla h_j(x^*) = 0$ para las restricciones de desigualdad, lo siguiente es verdadero:

$$(6') \quad V^T \nabla^2 L \quad (x^*, u^*, v^*) \quad V \qquad 0$$

Ejemplo 1. Condiciones necesarias y suficientes con restricciones de desigualdad.

Minimizar: $f(x) = x_1^2 + x_2$

Sujeta a: $g_1(x) = (x_1^2 + x_2^2) + q \geq 0$

$g_2(x) = -x_1 - x_2 + L \geq 0$

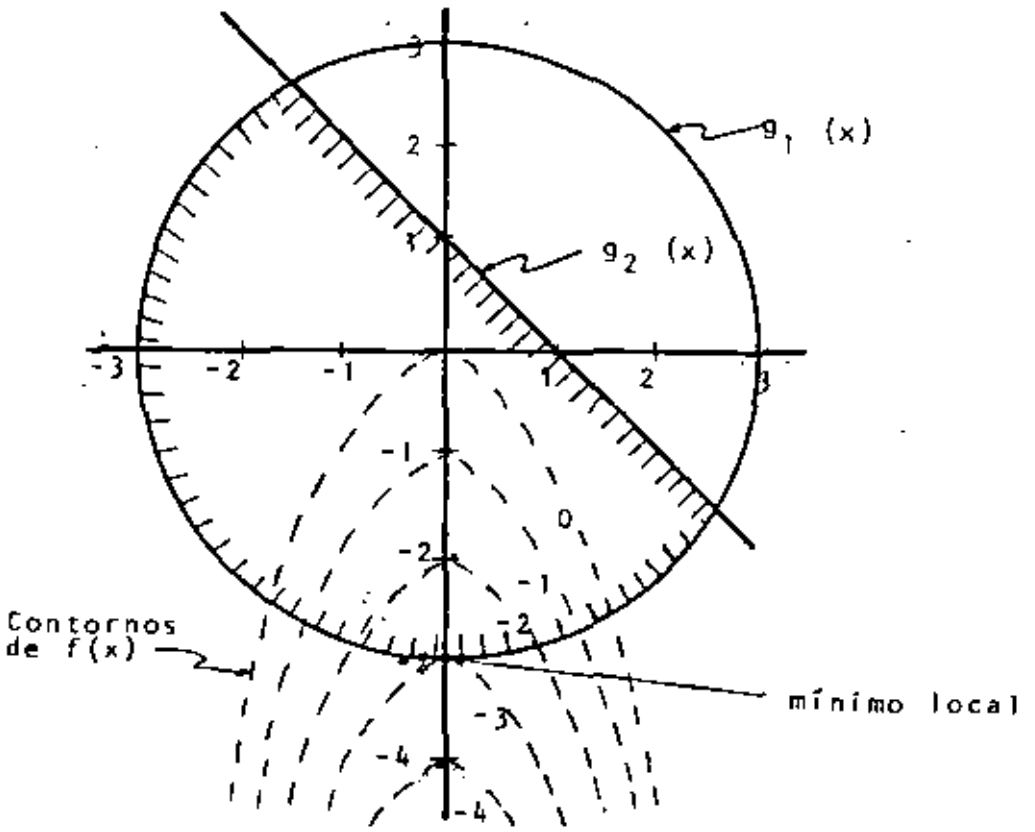


FIGURA 4. Región admisible y curvas de contorno del Ejemplo 1. Se observa que $g_1(x)$ es una restricción activa mientras que $g_2(x)$ no lo es.

Ya que sólo una de las restricciones está activa, no se necesita comprobar la,calificación a primer - y segundo,orden de las restricciones (Note que $f(x)$, $g_1(x)$ y $g_2(x)$ son dos veces diferenciables).

De acuerdo a los Teoremas (1) y (2) se necesita demostrar que existen u^* y x^* tales que

(2) $g_j(x^*) \geq 0$

$$-x_1^{*2} - s_2^{*2} + q \geq 0$$

$$-x_1^* - x_1^* + 1 \geq 0$$

(3) $u_j^* g_j(x^*) = 0$

$$u_1^*(-x_1^{*2} - x_2^{*2} - q) = 0$$

$$u_2^*(-x_1^* - x_2^* + 1) = 0$$

(4) $u_j^* \geq 0$

$$u_1^* \geq 0$$

$$u_2^* \geq 0$$

(5) $\nabla L(x^*, u^*) = 0$ $\quad (L = f(x) - u_1 g_1(x) - u_2 g_2(x))$

$$\begin{pmatrix} 2x_1^* \\ 1 \end{pmatrix} - u_1^* \begin{pmatrix} -2x_1^* \\ -2x_2^* \end{pmatrix} - u_2^* \begin{pmatrix} -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Resolviendo las ecuaciones anteriores se puede verificar que :

$$x^* = (0, -3)^T$$

y $$u^* = \left(\frac{1}{6}, 0\right)^T$$

La condición de segundo orden que debe ser satisfecha es :

$$V^T \nabla g_j(x^*) = 0 \qquad g_j(x^*) : \text{restricción activa}$$

es decir: $$(V_1, V_2) \begin{pmatrix} -2x_1^* \\ -2x_2^* \end{pmatrix} = V_1(0) + V_2(6) = 0$$

de donde V_1 puede tomar cualquier valor, y $V_2 = 0$, substituyendo v en (6), se tiene

(6) $$V^T \nabla^2 L(x^*, u^*) V \geq 0$$

donde $$\nabla^2 L = \begin{pmatrix} 2(1+u_1) & 0 \\ 0 & 2u_2 \end{pmatrix}$$

Ejemplo 2. Condiciones necesarias y suficientes con restricciones de Igualdad y de Desigualdad

Minimize : $$f(x) = x_1^2 + x_2$$

Sujeta a : $$h_1(x) = x_1^2 + x_2^2 - 9 = 0$$

$$g_2(x) = -(x_1 + x_2^2) + 1 \geq 0$$

$$g_3(x) = -(x_1 + x_2) + 1 \geq 0$$

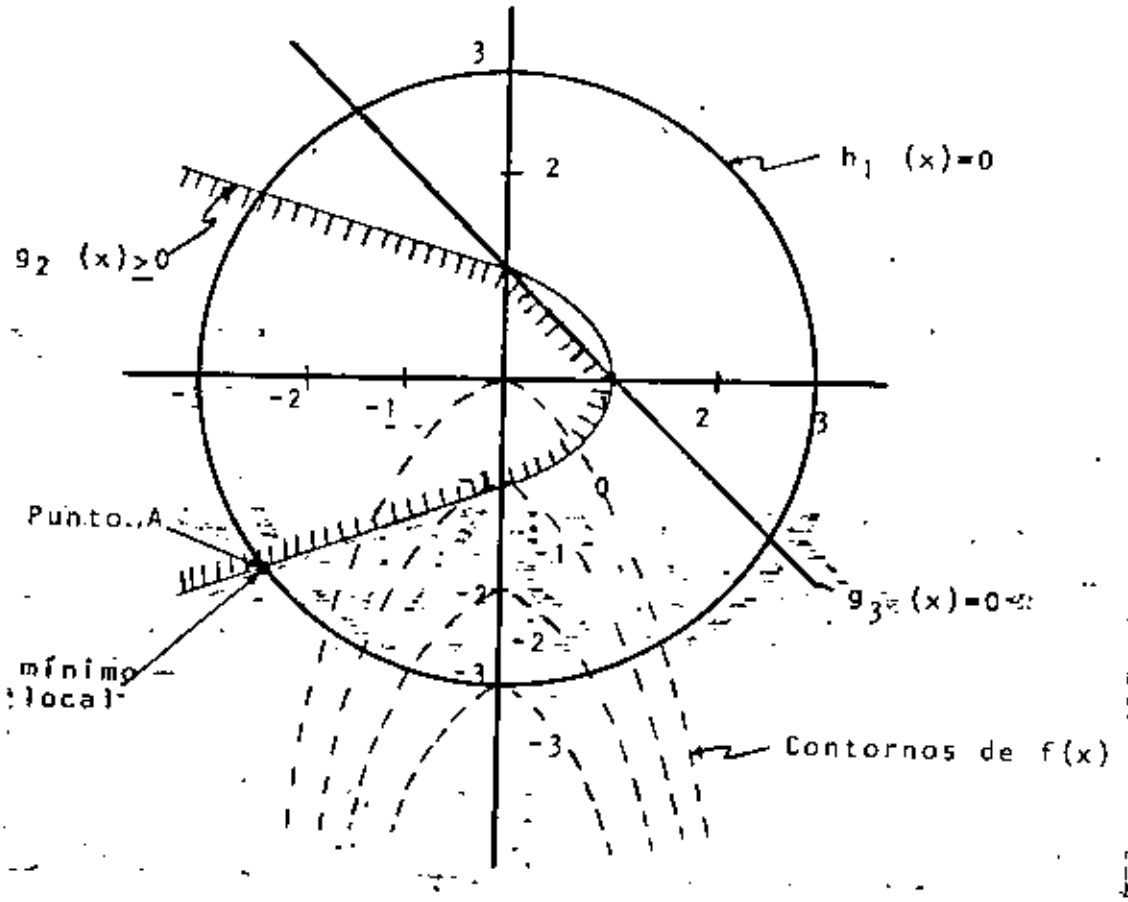


FIGURA 5. Región admisible y curvas de nivel del Ejemplo 2.

De acuerdo al Teorema 3, $h_1(x)$, $g_2(x)$ y $g_3(x)$ deben ser dos veces diferenciables. Además, los gradientes de las restricciones activas deberán ser linealmente independientes para que satisfagan la calificación a primer y segundo orden. Suponga que A, localizado en $x^* = (-2.37, -1.84)^T$ en la intersección de $h(x^*) = g_2(x^*) = 0$ sea un candidato a mínimo local. Si se forma una combinación lineal entre los gradientes de $h_1(x)$ y $g_2(x)$, en x^*, se tiene que, para que sean

linealmente independientes

$$C_1 \nabla h_1(x^*) + C_2 \nabla g_2(x^*) = 0 \qquad C_1 \text{ y/o } C_2 = 0$$

o

$$C_1 \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} + C_2 \begin{bmatrix} -1 \\ -2x_2^* \end{bmatrix} = 0$$

y como $\det \begin{bmatrix} 2x_1^* & -1 \\ 2x_2^* & -2x_2^* \end{bmatrix} \neq 0 \qquad C_1 = C_2 = 0$

es decir, los gradientes de las restricciones activas son linealmente independientes.

De acuerdo a los Teoremas (1) y (2) se debe cumplir que

(1) $h_1(x^*) = 0 \qquad x_1^{*2} + x_2^{*2} = 0$

(2) $g_j(x^*) = 0 \qquad -(x_1^{*2} + x_2^{*2}) + 9 \geq 0$

$$-(x_1^* + x_2^*) + 1 \geq 0$$

(3) $u_j^* g_j(x^*) = 0 \qquad u_2^* \left[-(x_1^{*2} + x_2^{*2}) + 9 \right] = 0$

$$u_3^* \left[-(x_1^* + x_2^*) + 1 \right] = 0$$

(4) $u_j^* \geq 0 \qquad u_2^* \geq 0$

$$u_3^* \geq 0$$

(5) $\nabla L(x^*, u^*, w^*) = 0 \qquad L = f(x^*) + w_1^* h_1(x^*) - u_2^* g_2(x^*) - u_3^* g_3(x^*)$

$$\begin{pmatrix} 2x_1^* \\ \\ 1 \end{pmatrix} + W_1^* \begin{pmatrix} 2x_1^* \\ \\ 2x_2^* \end{pmatrix} - u_2^* \begin{pmatrix} -1 \\ \\ -2x_2^* \end{pmatrix} - u_3^* \begin{pmatrix} -1 \\ \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ \\ 0 \end{pmatrix}$$

de donde se obtiene que:

$$w_1^* = -0,779$$

$$u_1^* = 1,05$$

$$u_3^* = 0$$

De acuerdo al Teorema 3, se debe encontrar V tal que:

$V^T \nabla g_j(x^*) = 0$ para las restricciones de desigualdad activas, y además $V^T \nabla h_1(x^*) = 0$, es decir

$$(V_1, V_2) \begin{pmatrix} -1 \\ \\ -2x_2^* \end{pmatrix} = 0 \qquad V_1 + 2x_2^* V_2 = 0$$

$$V_1, V_2 \begin{pmatrix} 2x_1^* \\ \\ 2x_2^* \end{pmatrix} = 0 \qquad 2x_1^* V_1 + 2x_2^* V_2 = 0$$

de donde se obtiene que $V = (v_1, v_2)^T = (0, 0)^T$ y entonces existe una única solución al problema en la intersección de $g_2(x)$ y $h_1(x)$.

Note que en este problema en particular

$$V^T \nabla^2_L (x^*, u^*, w^*) V \geq 0$$

para toda $V \neq 0$, es decir $\nabla^2 L(x^*, u^*, w^*)$ es una matriz positiva definida. [condición (6')].

Los dos ejemplos analizados en este capítulo tienen el propósito de ilustrar las condiciones de optimalidad de primero y segundo orden. Ahora bien, los problemas que se pueden tratar analíticamente son demasiado sencillos, sin que esto quiera decir que para problemas muy grandes y/o complicados lo anterior no sea válido.

B I B L I O G R A F I A

1. A.V. Fiacco and G.P. Mc Cormick. "Non linear Programming", John Wiley and Sons, Inc., New York, 1968.

2. W.W. Kuhn and A. W. Tucker, "Non linear Programming. Proc. 2nd. Berdeley Symp. on Mathematical Statistics and Probability, Univ. of Calif. Press, Berkeley, 1951, pp. 481-492.

3. G.P. Mc. Cormick. SIAM J. Appl. Math., 15 : 641 (1967)

4. O.L. Mangasarian, "Non linear Programming", Mc. Graw-Hill, New York, 1969.

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

OPTIMACION DE SISTEMAS FISICOS

METODOS DE OPTIMACION SIN RESTRICCIONES

M. EN C. SUSANA GOMEZ GOMEZ

MARZO, 1981

METODOLOGIA

I. BUSQUEDAS UNIDIRECCIONALES

Casi la totalidad de las técnicas de minimización que se describirán en otras secciones más adelante, requieren de técnicas de minimización unidimensionales las cuales tienen como propósito el localizar el mínimo local de una función de una variable. La implementación de los métodos mencionados requieren del conocimiento previo de un cierto intervalo cual contiene al mínimo de la función objetivo $f(x)$, y además se supone que en el intervalo prescrito la función es unimodal. En la Tabla II se mencionan algunos de los métodos más conocidos para lograr la minimización deseada, todos los cuales tienen como propósito reducir el tamaño del intervalo $\Delta^{(o)}$ hasta un tamaño $\Delta^{(n)}$. Para comparar la rapidez relativa de los métodos, Wilde define una eficiencia para n evaluaciones de la función como :

$$\text{Eficiencia} = L = \frac{\Delta^{(n)}}{\Delta^{(o)}}$$

En la tabla I se comparan los valores de E para varios métodos.

Métodos no-secuenciales	δ	Secuenciales	E
Búsqueda uniforme	$\frac{2}{n+1}$	Búsqueda secuencial (Dicotomus)	$\frac{1}{2^{n/2}}$
Búsqueda uniforme (Dicotomus)	$\frac{1}{\frac{n}{2}+1}$	Búsqueda Fibonacci	$\frac{1}{F_n^*}$
		Sección Dorada	$(0.618)^{1-n}$

(*) : Número de Fibonacci para n evaluaciones

TABLA II. Eficiencia de Técnicas de Búsqueda Unidimensional

En la Tabla III se compara el número de evaluaciones de la función que se requiere para reducir un intervalo inicial de 5×15^{1} a uno menor.

	NO SECUENCIAL		SECUENCIAL		
$\Delta^{(n)}$	Uniforme	Dicotomus	Dicotomus	Fibonacci	Sección Dorada
$5 * 10^{-3}$	199	198	14	11	11
$5 * 10^{-5}$	19,999	19,998	28	21	21

TABLA III. Número de evaluación de la función para reducir $\Delta^{(o)} = 6 \times 10^{-1}$

A continuación se describe el método de la sección dorada. Este método esta basado en la división de una línea en dos segmentos tales que la relación del tamaño original de la linea al segmento mayor es la misma

que la relación del segmento mayor al menor, es decir :

$$F_1 + F_2 = 1$$

$$\frac{1}{F_2} \quad \frac{F_2}{F_1} \quad : \quad F_2^2 = F_1$$

de donde

$$F_1 = \frac{3 - \sqrt{5}}{2} = 0.38$$

$$F_2 = \frac{\sqrt{5} - 1}{2} = 0,62$$

Para iniciar la búsqueda del mínimo de $f(x)$, se necesita especificar (o averiguar) en que dirección ésta se llevará a cabo. (Se supondrá que se conoce).

Como primer paso se debe encontrar un intervalo Δ donde se encuentra el mínimo de $f(x)$ usando, por ejemplo, una serie de pasos cada vez más grande sobre la variable independiente. Suponga que esto se ha hecho y que los últimos tres puntos obtenidos en x son los siguientes : $x_3^{(o)}$, el último, $x_2^{(o)}$ y $x_1^{(o)}$, donde $f(x_3^{(o)}) \geq f(x_2^{(o)})$ y sea $\Delta^{(k)} = x_3^{(k)} - x_1^{(k)}$ (Fig. 6). Lo anterior una vez hecho la k-ésima etapa modifica el intervalo de la siguiente forma.

$$y_1^{(k)} = x_1^{(k)} + F_1 \Delta^{(k)}$$

$$y_2^{(k)} = x_1^{(k)} + F_2 \Delta^{(k)} = x_3^{(k)} - F_1 \Delta^{(k)}$$

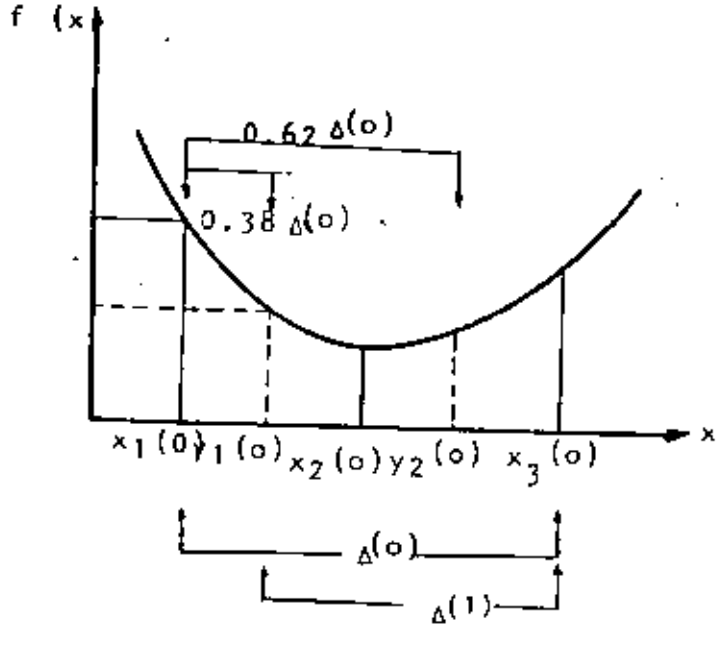
si $f(y_1^{(k)}) < f(y_2^{(k)})$: $\Delta^{(k+1)} = (y_2^{(k)} - x_1^{(k)})$, y $x_1^{(k+1)} = x_1^{(k)}$, $x_3^{(k+1)} = y_3^{(k)}$

si $f(y_1^{(k)}) > f(y_2^{(k)})$: $\Delta^{(k+1)} = (x_3^{(k)} - y_1^{(k)})$, y $x_1^{(k+1)} = y_1^{(k)}$, $x_3^{(k+1)} = x_3^{(k)}$

si $f(y_1^{(k)}) = f(y_2^{(k)})$: $\Delta^{(k+1)} = (y_2^{(k)} - x_1^{(k)})$ ó $(x_3^{(k)} - y_1^{(k)})$, y

$x_1^{(k+1)} = x_1^{(k)}$, $(x_3^{(k+1)} = y_2^{(k)}$, ó

$x_1^{(k+1)} = y_1^{(k)}$, $x_3^{(k+1)} = x_3^{(k)}$



$$y_1^{(o)} = x_1^{(o)} + 0.38 \Delta^{(o)}$$

$$y_2^{(o)} = x_1^{(o)} + 3.62 \Delta^{(o)}$$

FIGURA 6. Búsqueda mediante sección Dorada.

Otra clase de métodos para búsquedas unidireccionales. Localizar un punto x, cercano a x^* (el mínimo) mediante interpolación y extrapolación. En estos métodos se usan interpolaciones cuadráticas y cúbicas para aproximar el valor de la función. A continuación se describen un par de algoritmos que al usarlos en forma conjunta generan un algoritmo bastante poderoso para la localización de mínimos locales en una dirección: estos dos algoritmos son los siguientes:

a) Davis-Swann.- Compey (DSC) para definir el intervalo Δ donde se encuentra el mínimo, y

b) Powell, para definir la localización "exacta" del mínimo.

En el método DSC, se toman pasos de tamaño cada vez mayor hasta que se sobrepasa la localización del mínimo. A partir de ese momento se usa interpolación cuadrática (Figura 7). Los pasos que se siguen en este algoritmo son los siguientes

1. Evaluar $f(x)$ en el punto inicial $x^{(o)}$. Si $f(x^{(o)} + \Delta k$
   Si $f(x^{(o)} + \Delta x) \leq f(x^{o})$ se continua con el paso 2.
   Si $f(x^{(o)} + \Delta x) > f(x^{o})$ se define $\Delta x = -\Delta x$ (se cambia la dirección de búsqueda) y se continua con el paso 3.

2. $x^{(k+1)} = x^{(k)} + \Delta x$

3. Calcular $f(x^{(k+1)})$

4. Si $f(x^{(k+1)}) \leq f(x^{(k)})$, se duplica el tamaño de paso Δx y se repite el procedimiento desde el paso 2. Si $f(x^{(k+1)}) > f(x^{(k)})$ sea $x^{(m)} = x^{(k+1)}$, $x^{(m-1)} = x^{(k)}$, etc., reduzcase x a la mitad y regrese a los pasos 2 y 3 una vez más.

5. De los cuatro puntos igualmente espaciados x, en el conjunto $\{x^{(m+1)}, x^{(m)}, x^{(m-1)}, x^{(m-2)}\}$, se eliminan o $x^{(m)}$ o $x^{(m-2)}$, el que este más lejos de la x con el valor de la función más baja. Los tres puntos restantes se denotan como $x^{(o)}$, $x^{(b)}$ y $x^{(c)}$, donde $x^{(b)}$ es el punto central y $x^{(a)} = x^{(b)} - x$ y $x^{(c)} = x^{(b)} + \Delta x$.

6. Use interpolación cuadrática para estimar x^*,

$$x^* \approx \bar{x}^* = x^{(b)} + \frac{\Delta x \left[f(x^{(a)}) - f(x^{(c)})\right]}{2\left\{f(x^{(a)}) - 2f(x^{(b)}) + f(x^{(c)})\right\}}$$

Los pasos anteriores completan la primera etapa del método DSC. Para continuar, se reinicia en $\bar{x}^*$ o $x^{(c)}$, si $f(x^{(2)}) < f(\bar{x}^*)$, se reduce Δx y se empieza desde el paso 1 nuevamente.

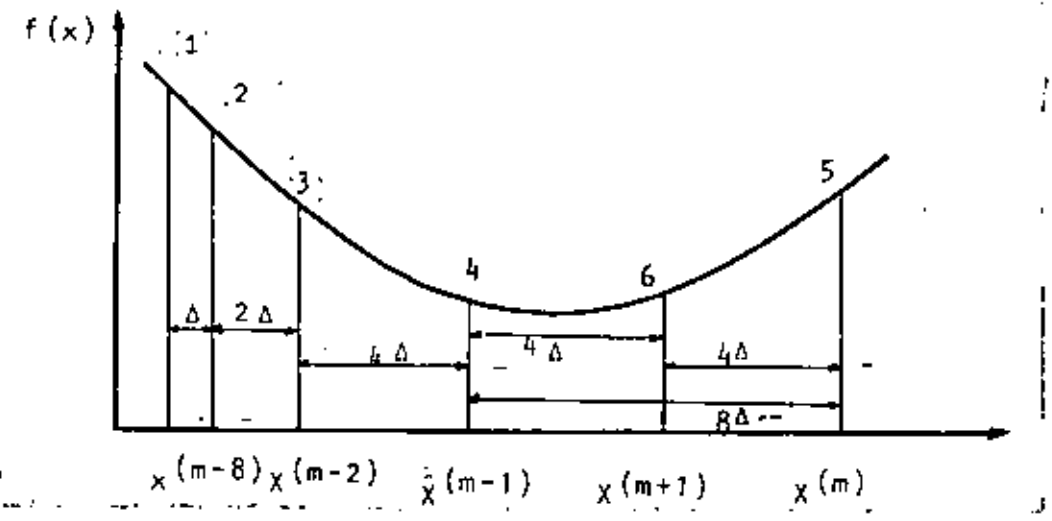


FIGURA 7. Método DSC para minimización unidimensional.

En el método de Powell, se usa una aproximación cuadrática usando los tres primeros puntos obtenidos en la dirección de búsqueda dada. La x correspondiente al mínimo de la función cuadrática se usa para efectuar una nueva aproximación y se continua de esta forma hasta localizar el mínimo de f (x) (Figura 8).

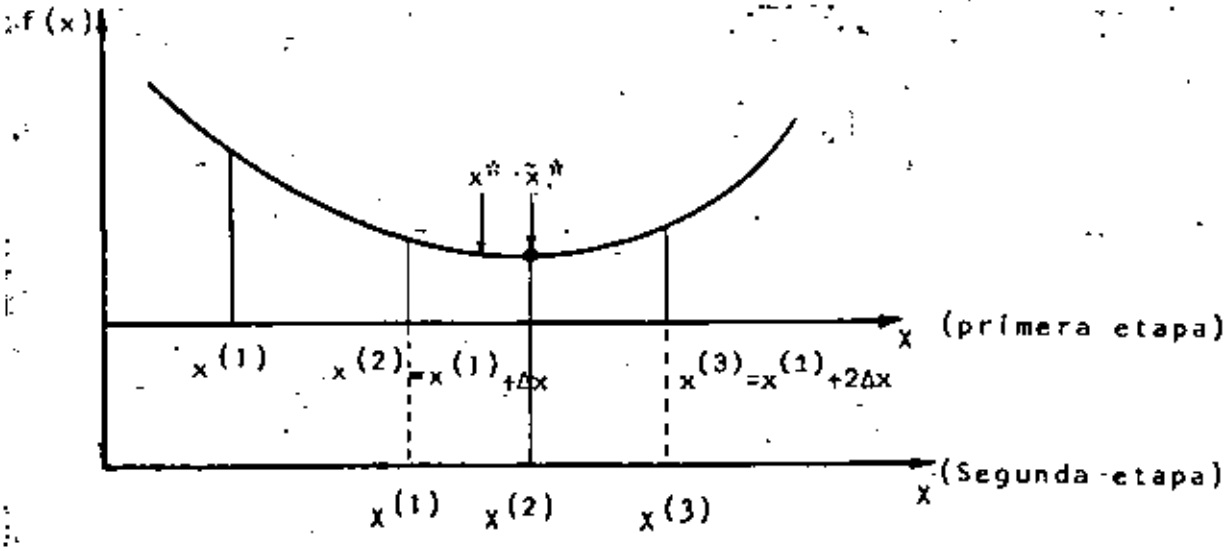


FIGURA 8. Método de Powell para minimización unidimensional.

Los pasos que deben seguirse para implementar el método de Powell son:

1. dados $x^{(1)}$ y Δ, calcule $x^{(2)} = x^{(1)} + \Delta$

2. calcule $f(x^{(1)})$ y $f(x^{(2)})$

3. Si $f(x^{(1)}) > f(x^{(2)})$, $x^{(3)} = x^{(1)} + 2\Delta$

   Si $f(x^{(1)}) \leq f(x^{(2)})$, $x^{(1)} = x^{(1)} - \Delta$

4. Calcule $f(x^{(1)})$

5. Estime el valor de x en el mínimo de $f(x)$, $\bar{x}^*$, como

$$x^* = \bar{x}^* = \frac{\left[(x^{(2)})^2 - (x^{(3)})^2\right] f(x^{(1)}) + \left[(x^{(\ )})^2 - (x^{(1)})^2\right] f(x^{(2)}) + \left[(x^{(1)})^2 - (x^{(2)})^2\right] f(x^{(3)})}{(x^{(2)} - x^{(3)}) f(x^{(1)}) + (x^{(3)} - x^{(1)}) f(x^{(2)}) + (x^{(1)} - x^{(2)}) f(x^{(3)})}$$

6. Si $\bar{x}^*$ o algunos de entre $\{x^{(1)}, x^{(2)}, x^{(3)}\}$ que corresponda al valor más pequeño de $f(x)$, difiere en menos que la exactitud presenta para x, o la exactitud en el valor de $f(x)$, se termina la búsqueda. En caso contrario, evalúe $f(\bar{x}^*)$ y se elimina del conjunto $\{x^{(1)}, x^{(2)}, x^{(3)}\}$ del que tenga el valor más alto de la función $f(x)$, a menos que se pierda el intervalo de x donde se encuentra el mínimo, en cuyo caso se debe eliminar una x tal que el intervalo adecuado no se pierda. Se repita el procedimiento desde el paso 6.

II. MINIMIZACION SIN RESTRICCIONES USANDO DERIVADAS

El problema general de minimización sin restricciones se puede plantear como :

$$\text{Minimizar} : \quad f(x) , \quad x \in L^{-n} \qquad (1)$$

donde $f(x)$ es la función objetivo. Según se vió en el capitulo anterior, se pretende encontrar un punto x^* tal que $\nabla f(x^*) = 0$. En la presente sección se atacará el problema definido en (1) mediante el uso de métodos que hagan uso de las primeras y segundas derivadas parciales $\left(\nabla f(x) \text{ y } \nabla^2 f(x)\right)$ de la función objetivo.

2.1. Método del Máximo Descenso (Gradiente)

Según se recordará del capitulo anterior, el gradiente de la función objetivo, $f(x)$, en cualquier punto, x, es un vector dirigido en la dirección del máximo incremento local en $f(x)$. Resulta obvio que se puede escoger como dirección de búsqueda, para minimizar $f(x)$, la dirección opuesta al gradiente, $-\nabla f(x)$, esto es, en la dirección de máximo descenso. Si se escoge como dirección de búsqueda la ya señalada, resultará lo siguiente.

a) sea $\nabla s_r = - f(x_k)$ la dirección de búsqueda en el k-ésimo paso del algoritmo.

b) Si $\Delta x_k = + \alpha_a S_k = - \alpha_k \nabla f(x_k)$, es el desplazamiento del punto x_k al x_{k+1}, es decir

$$x_{k+1} = x_k + \Delta x_k \qquad (\alpha_k \text{ un escalar positivo})$$

c) Entonces, la aproximación a primer orden de $f(x)$ quedará como

$$f(x_k + \Delta x_k) = f(x_k) + \nabla^T f(x_k) \Delta x_k$$

o bien

$$f(x_k + \Delta x_k) - f(x_k) = - \alpha_k \nabla^T f(x_k) \nabla f(x_k) < 0$$

es decir, se puede garantizar que para α_k suficientemente pequeño, el valor de la función en x_{k+1} decrecerá con respecto al valor previo en x_k, siempre y cuando $\nabla f(x_k) \neq 0$.

(En los puntos (a) - (c), α_k se le conoce como tamaño de paso).

Existen varias alternativas para escoger el tamaño de paso α_k de las cuales se pueden mencionar las siguientes

a) Fijar el valor de α_k de antemano e igual para todas las iteraciones del método. La desventaja de proceder de esta forma es que no se puede garantizar que de una iteración a

la siguiente, el valor de la función decrezca, ya que esta propiedad sólo es válida cuando $\alpha_k \longrightarrow 0$.

Debido a lo anterior, el método puede presentar las siguientes desventajas. En primer lugar, que si α_k no es lo "suficientemente pequeña" el método ascilará. Por otro lado, si α_k se escoge "demasiado pequeña" la convergencia puede volverse demasiado pequeña.

b) Una segunda alternativa es escoger α_k en cada iteración de forma tal que el valor de la función objetivo se reduzca para algún cierto valor de α_k. Para lograr lo anterior se puede proceder de la siguiente forma en cada etapa.

o) , se escoge α ref > 0 , $0 < \pi < 1$ un factor multiplicativo

i) se fija $\beta_o = \alpha$ref, $i = 0$

ii) $y^{(i)} = x_k + \beta_o \Delta_k$

iii) calcular $f(y^i)$

iv) si $f(y^i) < f(x_k)$ continuar con el paso vii)

v) $i \longleftarrow i + 1$

vi) $\rho_{i+1} = \pi\beta_i$; repetir el procedimiento desde (ii)

vii) $x_{k+1} = y^{(i)}$; $\alpha_k = \beta_i$

$f(x_{k+1}) = f(y^i)$

El procedimiento anterior garantiza que, si se permite un número ilimitado de iteraciones $\beta_{i+1} = \pi\beta_i$, en cada etapa del método de máximo descenso, se obtendrá un descenso en la función objetivo. Tiene el defecto a que puede consumir demasiado tiempo en la búsqueda de un tamaño de paso adecuado α_k.

c) Una tercera alternativa es la siguiente. Supongase que de alguna forma la dirección de búsqueda S_k, es decir, el nuevo iterando x_{k+1} caerá sobre la ecuación de un parámetro:

$$x_{k+1} = x_k + \alpha S_k$$

siendo α el parámetro. La diferencia entre este procedimiento y los dos anteriores es que en el presente se pide que el tamaño de paso α sea tal que $f(x_{k+1}) = f(x_k + \alpha S_k)$ adquiera su mínimo valor 0, formalmente

$$\frac{d f(x_k + \alpha S_k)}{d\alpha} = 0$$

Por ejemplo, si $f(x)$ es una función cuadrática

$$f(x) = a + b^T x + \tfrac{1}{2} x^T Q x \quad (Q: \text{positiva definida simétrica}).$$

entonces

$$S_k = -\nabla f(x_k) = -b - Qx_k$$

$$x_{k+1} = -\alpha_k (b + Q x_k)$$

$$\frac{\partial f(x_k - \alpha_k (b + Q x_k))}{\partial \alpha_k} = 0 = \nabla^T f(x_k) S_k + S_k^T Q (\alpha_k S_k)$$

de donde

$$\alpha_k = - \frac{\nabla^T f(x_k) S_k}{}$$

Si se supone que $f(x)$ no es una función cuadrática de x, entonces se pueden emplear cualquiera de las técnicas de búsquedas unidimensionales descritas en la sección anterior.

Una característica interesante de este procedimiento de minimización es que el gradiente en el nuevo punto, $x_k + \alpha f(x_{k+1})$, es ortogonal a la dirección de búsqueda empleada para localizar x_{k+1}, es decir

$\nabla^T f(x_{k+1}) S_k = 0$, lo cual se demuestra como sigue. Supóngase la misma función cuadrática ya empleada entonces

$$\nabla f(x_k) = b + Q x_k$$

y de la expresión para $\frac{d f(\alpha)}{d\alpha} = 0$ se obtiene

$$(b + Q x_k)^T S_k + S_k^T Q \alpha_k S_k = 0$$

y como $\alpha x_{k+1} - x_k = \alpha_k s_k$, entonces

$$s_k^T (b + Q x_k) + s_k^T Q (x_{k+1} - x_k) = 0.$$

o bien

$$s_k^T (b + H x_{k+1}) = s_k^T \nabla f(x_{k+1}) = 0.$$

Lo anterior se ilustra en la Figura 9.

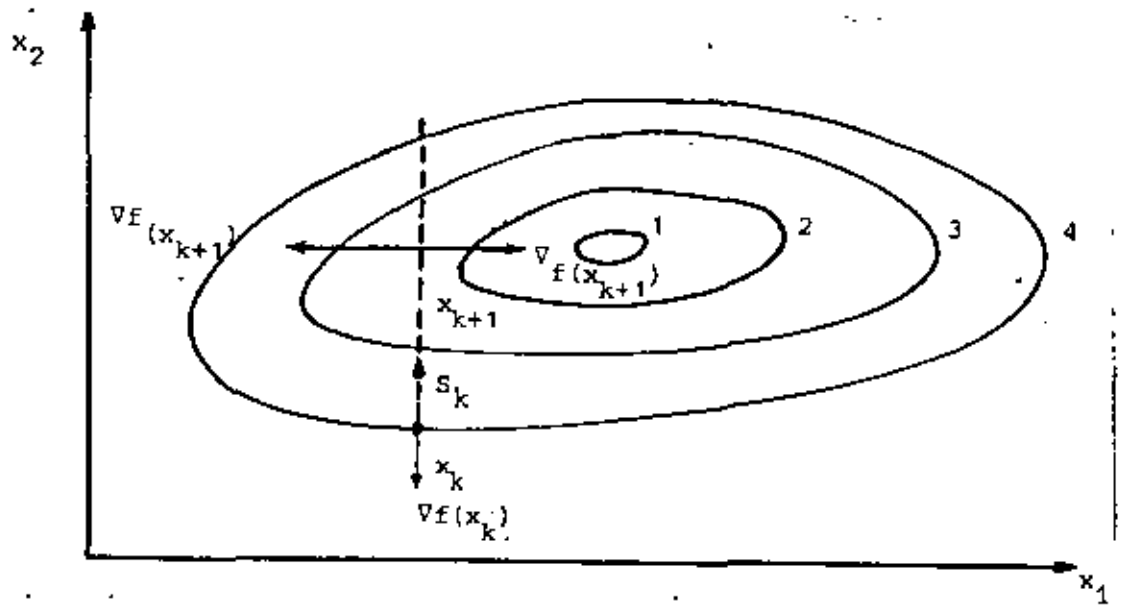


FIGURA 9. Ortogonalidad de las direcciones de Búsqueda

La implementación práctica del algoritmo de máximo descenso, con cualquiera de las tres alternativas discutidas para determinar α_k, quedaría de la siguiente forma.

1. $k = 0$

2. Estimar x_o (punto de arranque).

3. Calcular $f(x_k)$, $\nabla f(x_k)$

4. Si $\nabla^T f(x_k)\ \nabla f(x_k) \leq$ tolerancia, se ha encontrado un punto estacionario de la función $f(x)$

5. $S_k = -\nabla f(x_k)$

6. Cálculo de α_k (ver texto)
   como resultado se calcula x_{k+1} y $\nabla f(x_{k+1})$

7. Calcular $\nabla f(x_{k+1})$.

8. $k \longrightarrow k + 1$; se repite el procedimiento desde el paso 4.

Para finalizar la discusión sobre el método del máximo descenso es conveniente hacer notar que bajo algunas condiciones, por cierto no muy frecuentes, el algoritmo puede ser atraido por un punto silla ya que también en esta clase de puntos se satisface que $\nabla^T f(x)\ \nabla f(x) = 0$, no existiendo forma de detectar a priori que tal cosa sucederá. Ahora bien, para clasificar el tipo de punto donde se detuvo el algoritmo, es necesario analizar la matriz de segundas derivadas, de acuerdo a :

i) H, positiva definida — mínimo

ii) H, negativa definida — máximo

iii) H, semi-positiva o semi-negativa definida — punto silla.

2.2. Método de Newton

El método de Newton que a continuación se presenta esta basado en una aproximación a segundo orden de la función objetivo $f(x)$, es decir, usa información de segundo orden (matriz hessiana), de aqui que se le clasifique como método de segundo orden.

Considérese la aproximación a segundo orden de $f(x)$

$$f(x + \Delta x) = f(x) + \nabla^T f(x)\,\Delta x + \tfrac{1}{2}\,\Delta x^T \nabla^2 f(x)\,\Delta x$$

Si se supone que la aproximación anterior es buena, como de hecho lo es en la vecindad de un mínimo y de un máximo, entonces, al derivar parcialmente $f(x + \Delta x)$ con respecto a cada uno de los elementos de Δx se obtiene

$$\frac{\partial f(x+\Delta k)}{\partial \nabla_x} = \nabla f(x) + \nabla^2 f(x)\Delta x = 0$$

de donde

$$\Delta X = - H^{-1}(x)\nabla f(x)$$

donde $H(x) = \nabla^2 f(x)$ es la matriz hessiana $\left(hij = \dfrac{\partial^2 f(x)}{\partial x_i \partial x_j}\right.$ y ΔX será la dirección del desplazamiento desde un punto x_k a x_{k+1}, es decir

$$X_{k+1} = X_R - H^{-1}(X_k)\nabla f(x_k)$$

El empleo de la última fórmula para generar los iterandos del método de Newton puede provocar los siguientes problemas si es que el método se está empleando para minimizar una función $f(x)$. El problema es el siguiente.. Note que tanto un máximo como un mínimo, asi como los puntos silla, satisfacen $\partial f(x + \Delta x)/ \partial \Delta x = 0$, . por lo que no se puede garantizar que el método converga a un mínimo, si no se modifica adecuadamente Esta modificación consiste en lo siguiente :

sea $\Delta x_k = - H^{-1}(x_k) \nabla f(x_k)$ la dirección de avance del punto x_k al x_{k+b} y considérase la aproximación a primer orden de $f(x)$ como sigue :

$$f(x_k + \Delta x_k) = f(x_k) + \alpha_k (\nabla^T f(x_k)\, \Delta x_k) \rho_k$$

o bien

$$\Delta f_k = f(x_k + \Delta x_k) - f(x_k) = \alpha_r (\nabla^T f(x_k)\, \Delta x_k)\, \rho k$$

donde α_k es el tamaño de paso, descrito en el inciso anterior, y ρ_k es el sentido de la dirección de búsqueda, el cual se determina como sigue : ya que se desea que $\Delta f_k < 0$ ($f(x_{k+1}) < f(x_k)$ 'y como $\alpha_k > 0$, entonces

$$\rho_k = \begin{cases} 1 & \text{si } \nabla^T f(x_k) H^{-1}(x_k) \nabla f(x_k) > 0 \\ -1 & \text{si } \nabla^T f(x_k) H^{-1}(x_k) \nabla f(x_k) < 0 \end{cases}$$

Y con esto se garantiza que si $\alpha_k (>o)$ es suficientemente pequeña entonces $f(x_{k+1}) < f(x_k)$ y el método de Newton convergerá a un mínimo, o en el peor de los casos a un punto silla, pero nunca a un máximo (se omite la discusión sobre el tamaño de paso α_k, por ser idéntica a la presentada con anterioridad).

Tomando en cuenta los puntos anteriores, la implementación del método de Newton queda como sigue

1. $k = 0$
2. Estimar x_o (punto de arranque)
3. Calcular $f(x)$,
4. Calcular $\nabla f(x_k)$
5. Si $\nabla^T f(x_k) \nabla f(x_k) \leq$ tolerancia, se ha encontrado un punto estacionario de $f(x)$ (mínimo o punto silla).
6. Calcular $H(x_k)$ y $H^{-1}(x_k)$
7. $$\rho_k = \begin{cases} 1 & \text{si} \quad \nabla^T f(x_k) \; H^{-1}(x_k) \; \nabla f(x_k) > 0 \\ -1 & \text{si} \quad \nabla^T f(x_k) \; H^{-1}(x_k) \; \nabla f(x_k) < 0 \end{cases}$$
8. $\Delta x_k = - \rho k \, H^{-1}(x_k) \nabla f(x_k)$
9. Calcular α_k (inciso 2.1)

   como resultado se obtiene x_{k+1}, $f(x_{k+1})$.

10. $k \longleftarrow k + 1$. Se repite el procedimiento desde (4).

Es fácil ver que si la función objetivo es cuadrática, p. ej.

$$f(x) = a + b^T x + \tfrac{1}{2} x^T Q x$$

el método de Newton converge en una sola iteración, ya que

$$\nabla f(x_k) = b + Q x_k$$

y $$\nabla^2 f(x_k) = Q$$

entonces

$$x_{k+1} = x_k - \left(\nabla^2 f(x)\right)^{-1} \cdot \nabla f(x_k) = x_k - Q^{-1}(b + Q x_k)$$

$$\boxed{Q^{-1} b = x^*}$$ convergencia cuadrática

2.3. Conjugancia y Direcciones Conjugadas

Como se verá más adelante, una función objetivo cuadrática de n-variables que exhibe un mínimo, puede ser minimizada en n pasos (o menos) si estos pasos se toman en lo que se ha llamado direcciones conjugadas. En el inciso siguiente se limitará la discusión a funciones cuadráticas del tipo

$$f(x) = a + b^T x + \tfrac{1}{2} x^T H x \qquad (o)$$

donde H es una matriz positiva definida.

2.3.1. Conjugancia

Supóngase que la minimización de $f(x)$ empieza en x_o en la dirección $\bar{S}^o$, escogida arbitrariamente (o por algún algoritmo); se supondrá $(\bar{S}_o^o)^T \bar{S}_o^o = 1.0$. El siguiente punto generado por el algoritmo será

$$x_1 = x_o + \lambda_o \bar{S}^o \qquad (1)$$

donde el tamaño de paso λ_o se determina minimizando $f(x_o + \lambda_o \bar{S}_o$ con respecto a λ, es decir

$$\frac{df(x_o + \lambda_o \bar{S}_o)}{d\lambda} = 0 = \nabla^T f(x_o)\bar{S}_o + (\bar{S}_o)^T \nabla^2 f(x_o)\left[\bar{S}_o \lambda\right]$$

de donde

$$\lambda_o = - \frac{\nabla^T f(x_o)\bar{S}_o}{\bar{S}_o^T \nabla^2 f(x_o)\bar{S}_o} \qquad (2)$$

Una vez que se encuentra el siguiente iterando, x_1, se deberá seleccionar una nueva dirección de búsqueda para la minimización de $f(x)$. La nueva dirección $\bar{S}_1$ se dice que es conjugada con $\bar{S}_o$ si $(\bar{S}_1)^T \nabla^2 f(x_o)\bar{S}_o = 0$. (En general, un conjunto de n direcciones independientes de búsqueda s_o, s_1, ... S_{n-1} son conjugadas con respecto a una matriz Q, positiva definida, si

$$S_i^T Q S_j = 0 \qquad 0 \leq i \neq j \leq n-1 \qquad (3)$$

Q, podría ser, por ejemplo, la matriz hessiana de la función objetivo H. Note además que si $Q = I$, la matriz unitaria, conjugancia y ortogonalidad son sinónimos).

Debido a que los vectores s_i, son linealmente independientes además de conjugados, cualquier vector $V \in E^n$, se puede representar en términos de aquellos, como

$$V = \sum_{j=0}^{n-1} V_j S_j$$

$$V_j = \frac{S_j^T H(x) V}{S_j^T H(x) S_j}$$

Otra relación importante que se utilizará más adelante es la siguiente. Considerese la matriz P definida por

$$P = \sum_{j=0}^{n-1} \alpha_j S_j S_j^T$$

Resulta obvio que si las α_j se escogan de manera tal que

$$P H s_k = s_k ,$$

entonces $P = H^{-1}$. Ahora bien.

$$P\,H\,S_k = \left(\sum_{j=0}^{n-1} a_j S_j S_j^T\right) H S_k = \sum_{j=0}^{n-1} a_j S_j (S_j^T H S_k)$$

$$= a_k S_k \; (S_k^T H S_k)$$

de donde si $a_k = (S_k^T H S_k)^{-1}$, entonces $P = H^{-1}$,

es decir

$$H^{-1} = \sum_{j=0}^{n-1} \frac{S_j S_j^T}{S_j^T H S_j}$$

si las direcciones de búsqueda empleadas en la minimización de $f(x)$ se escogen conjugadas (esto se demostrará más adelante) a continuación se demuestra que : <u>cualquier función cuadrática de n variables que exhibe un mínimo, puede ser minimizada en n pasos, si se emplean direcciones conjugadas</u>, una dirección diferente en cada paso. Además, el orden en que se usan las direcciones de búsqueda es irrelevante para alcanzar el mínimo.

<u>Demonstración</u> Sea $f(x) = a + b^T x + \frac{1}{2} x^T H x$, $\nabla f(x) = b + Hx$, $\nabla f(x) = H$, y en el mínimo de x, $\nabla f(x^*) = 0$, es decir $x^* = -H^{-1} b$

Para la n-ésima etapa se tiene, usando (1) y (2), que

$$x_n^n = x_o + \sum_{k=o}^{n-1} \lambda_k \bar{S}_k$$

y como en cada etapa se usó el valor óptimo de , dado por la ecuación (2),

$$x_n = x_o - \sum_{k=o}^{n-1} \frac{(\bar{S}_k)^T f(x_k)}{(\bar{S}_k)^T H \bar{S}_k} \bar{S}_k \qquad (5a)$$

Por otro lado

$$(\bar{S}_k)^T \nabla f(x_k) = (\bar{S}_k)^T (Hx_k + b)$$

$$= (\bar{S}_k)^T \left[H\left(X_o + \sum_{i=1}^{n-1} \lambda_i \bar{S}_i\right) + b \right]$$

$$= (\bar{S}_k)^T (Hx_o + b) = 0 \text{ por conjuganciа de los } \bar{S}_i$$

Entonces

$$x_n = x_o - \sum_{k=o}^{n-1} \frac{(\bar{S}_k)^T (Hx_o + b) \bar{S}_k}{(\bar{S}_k)^T H \bar{S}_k} \qquad (5.b)$$

Usando la relación (4.a) se obtiene que

$$X_o = \sum_{k=o}^{n-1} \frac{(\bar{S}_k)^T H x_o}{(\bar{S}_k)^T H \bar{S}_k} \bar{S}_k$$

y por lo tanto

$$x_n = \sum_{k=o}^{n-1} \frac{(\bar{S}_k)^T b \bar{S}_k}{(\bar{S}_k)^T H \bar{S}_k} = \sum_{k=o}^{n-1} \frac{(\bar{S}_k)^T H (H^{-1} b) \bar{S}_k}{(\bar{S}_k)^T H \bar{S}_k}$$

y usando (4 a) nuevamente, se obtiene

$$x_n = - H^{-1} b \qquad \text{f.e.d.}$$

Un método para el cual se garantiza que alcanza el mínimo de una función objetivo cuadrática en un número específico de pasos, se dice que tiene la propiedad de terminación cuadrática. (El método de gradiente conjugado necesita de n pasos, mientras que el de Newton uno sólo).

2.3.2. Método de Gradiente conjugado

El método de Fletcher - Reeves de gradiente conjugado, que a continuación se describe, genera una secuencia de direcciones de búsqueda que son combinación lineal de $- \nabla f(x_k)$, la dirección de máximo descenso en el último punto, y de las k direcciones de búsqueda anteriores, s_o, s_1, ..., s_{k-1}, usando factores de peso tales que S_k sea conjugada a las direcciones anteriores.

Para ilustrar el método, sea $S_o = - \nabla f(x_o)$, y

$x_1 = x_o + \lambda_o^* S_o$; sea

$$S_1 = - \nabla f(x_1) + \alpha_1 S_o \qquad (6)$$

donde α_1 se escoge de forma tal que S_o y S_1 sean conjugadas con respecto a H, es decir

$$S_o^T H S_1 = 0 \qquad (7)$$

Para eliminar S_o, considérese la aproximación a primer orden del gradiente, es decir

$$\nabla f(x_1) - \nabla f(x_o) = \nabla^2 f(x_o)(x_1 - x_o) = \lambda_o^* H S_o \tag{8}$$

$$S_o = \frac{1}{\lambda_o^*} H^{-1}\left[\nabla f(x_1) \nabla f(x_o)\right]$$

y como H es simétrica, entonces

$$S_o^T = \frac{\left[\nabla f(x_1) - \nabla f(x_o)\right]^T H^{-1}}{\lambda_o^*} \tag{9}$$

Substituyendo (6) y (9) en (7) se obtiene

$$\left[\nabla f(x_1) - \nabla f(x_o)\right]^T \left[-\nabla f(x_1) + \alpha_1 S_o\right] = 0 \tag{10}$$

Y ya que, según se vió con anterioridad $\nabla^T f(x_o) \nabla f(x_1) = \nabla^T f(x_1) S_o = 0$, entonces

$$\alpha_1 = -\frac{\nabla^T f(x_1) f(x_1)}{\nabla^T f(x_o) S_o}$$

o bien,

$$\alpha_1 = -\frac{\nabla^T f(x_1) \nabla f(x_1)}{\nabla^T f(x_o) \nabla f(x_o)} \tag{11}$$

La dirección de búsqueda S_2 se forma como una combinación lineal de $-\nabla f(k_2)$, S_1 y S_o, y se fuerza a que sea conjugada a S_1 y S_o

con lo que se obtiene la siguiente expresión para los factores de peso a_k

$$a_k = \frac{\nabla^T f(x_k)\ \nabla f(x_k)}{\nabla^T f(x_{k-1})\ \nabla f(x_{k-1})} \qquad (12)$$

La implementación del algoritmo de gradiente conjugado de Fletcher - Reeves incluye los siguientes pasos :

1. Estimar x_o (punto de arranque)

2. $S_o = -\nabla f(x_o)$

3. En la $k-$ésima etapa del algoritmo, se determina el mínimo unidireccional de $f(x)$, a lo largo de la dirección de búsqueda S_k. Con esto se localiza x_{k+1}.

4. La nueva dirección de búsqueda se determina como

$$S_{k+1} = \nabla f(x_{k+1}) + \frac{\nabla^T f(x_{k+1})\ \nabla f(x_{k+1})}{\nabla^T f(x_k)\ \nabla f(x_k)}\ S_k$$

Después de $(n+1)$ iteraciones $(k = n)$, se empieza un nuevo ciclo del algoritmo, es decir, x_{n+1} se convierte en x_o.

5. La búsqueda se da por terminada cuando en alguna iteración sucede que

$$S_k^T\ S_k \leq \text{tolerancia}$$

Note que al igual que en el método de gradiente ordinario, no se necesita la inversión de matriz alguna, lo cual es una ventaja.

2.4. Métodos de Métrica Variable

Los métodos de Métrica Variable o Cuasi-Newton son métodos que aproximan el hessiano, o su inversa, usando unicamente información acerca del gradiente. La mayoría de estos métodos usan direcciones conjugadas conforme avanzan, lo cual hacen siguiendo el esquema general

$$x_{k+1} = x_k + \lambda_k S_k = x_k - \alpha_k \eta(x_k) \nabla f(x_k) \qquad (1)$$

donde $\eta(x_k)$ representa una aproximación a $H^{-1}(x_k)$. (En el método de Gradiente ordinario $\eta(x_k) = I$, mientras que el método de Newton toma $\eta(x_k) = H^{-1}(x_k)$, con la desventaja de que hay que invertir el hessiano).

En una serie de métodos de Cuasi-Newton, $H^{-1}(x_{k+1})$ se aproxima de la información disponible en la k - ésima etapa, como

$$H^{-1}(x_{k+1}) \times \omega\eta_{k-1} = \omega(\eta_k + \Delta\eta_k) \qquad (2)$$

donde η es una aproximación a H^{-1}, $\Delta\eta_k$ es una matriz que se especifica de acuerdo al método, y ω una constante de escalamiento que frecuentemente se fija en 1. La selección de determina, esencialmente, el método

de método variable. Para garantizar convergencia, $\omega\eta_{k+1}$ debe ser positiva definida y debe satisfacer la siguiente relación cuando reemplaza a H^{-1}

$$x_{k+1} - x_k = H^{-1}(x_k)\left[\nabla f(x_{k+1}) - \nabla f(x_k)\right] \qquad (3\ a)$$

que es una aproximación a primer orden del gradiente.

En la (k^{+1}) - ésima etapa de cualquier método se conocen

x_k, $\nabla f(x_k)$ x_{k+1} $\nabla f(x_{k+1})$ y η_k, y se desea calcular η_{k+1}

De la relación obtenida con (2) y (3) como

$$\eta_{k+1}\Delta g_k = \frac{1}{\omega}\Delta x_k \qquad (3\ b)$$

donde $\Delta g_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. Sea $\Delta\eta_k = \eta_{k+1} - \eta_k$, por lo que la ecuación

$$\Delta\eta_k \Delta g_k = \frac{1}{\omega}\Delta x_k - \eta_k \Delta g_k \qquad (3\ c)$$

debe ser resuelta para $\Delta\eta_k$, y esta se obtiene como sigue. Si el lado derecho de (3 c) se multiplica y divide por $y^T \Delta g_k$, el primer término, y $z^T\Delta g_k$ el segundo término se obtiene que:

$$\left[\Delta\eta_k - \left(\frac{1}{\omega}\frac{\Delta x_k y^T}{y^T\Delta g_k} - \frac{\eta_k \Delta g_k z^T}{z^T\Delta g_k}\right)\right]\Delta g_k = 0$$

o bien,

$$\Delta\eta_k = -\left(\frac{1}{\omega}\right)\frac{\Delta x_k y^T}{y^T \Delta g_k} - \frac{\eta_k \Delta g_k}{Z^T \Delta g_k} Z^T \qquad (4)$$

donde los vectores columna y, Z son arbitrarios, al igual que w. Si por ejemplo se escogen

$$w = 1$$

$$y = Z = \Delta x_k - \eta_k \Delta g_k$$

se genera el algoritmo de Broyden, mientras que si se escogen

$$y = \Delta x_k$$

$$Z = \eta_k \Delta g_k$$

entonces la matriz η_{k+1} se actualiza de acuerdo al método de Davidon - Fletcher - Powell. Ya que los vectores y, Z son arbitrarios se pueden efectuar varias selecciones, las que se discuten más adelante. Si los pasos x_k se determinan mediante minimizaciones unidireccionales de $f(x)$ en la dirección S_k, todos los métodos que calculan una η_{k+1} simétrica que satisfagan (3 b), generan direcciones que son mutuamente ortogonales (para funciones cuadráticas).

2.4.1. $\Delta\eta_k$ de Rango 1

Broyden demostró que si $\Delta\eta_k$ es simétrica con rango 1, la relación $\eta_{k+1} \Delta g_k = \Delta x_k$ se satisface, la única posibilidad de escoger $\Delta\eta_k$ es

(5)

El algoritmo funcionaría de la siguiente forma. Se escogen x_o y $\eta_o > 0$ y se usa una forma secuencial (1), ( ) y (2) hasta que por ejemplo $\nabla^T f(x_k) \nabla f(x_k) \leq \varepsilon$. Por otro lado, si se usan minimizaciones unidireccionales, el método genera direcciones conjugadas y bajo algunas condiciones más o menos restrictivas, se puede demostrar que el algoritmo converge a la solución. Una característica atractiva de este método es que α_k en (1) no necesariamente tiene que ser un parámetro que minimize $f(x)$ a lo largo de S_k. El mismo Broyden demuestra que x puede tomar cualquier valor con la única condición de que no provoque que η se haga singular (denominador en (5)).

Si la función objetivo no es cuadrática, algunos de los aspectos poco satisfactorios al usar (5), son los siguientes :

1. puede dejar de ser positiva definida, en cuyo caso es necesario recurrir a alguna otra estrategia que lo garantice.

2. La corrección $\Delta\eta_k$ puede no quedar acotada (generalmente por errores de redondes, incluso para funciones cuadráticas)

3. Si por coincidencia $\Delta x_k = - \alpha_k \eta(x_k) \nabla f(x_k)$ queda en la dirección del paso anterior, $\eta(x^B_{k-1})$ se vuelve singular.

En consecuencia, en el algoritmo de Broyden, si sucede que

$$\eta_k \Delta g_k = \Delta x_k$$

o

$$(\eta_k \Delta g_k - \Delta x_k)^T \Delta g_k = 0$$

se fuerzo a que

$$\eta_{k-1} = \eta_k \qquad (\Delta \eta_k = 0)$$

2.4.2. Método de Davidon - Fletcher - Powell

En este método la matriz $\Delta\eta$ se escoge que tenga rango 2. La η inicial normalmente se toma como, $\eta = I$. (se puede usar cualquier otra matriz simétrica positiva definida), con lo que el método arranca con la dirección del máximo descenso. Conforme el método avanza, va existiendo un cambio del método de gradiente a Newton con lo que se obtiene una gran ventaja al usar las mejores características de ambos métodos.

Como se mencionó con anterioridad la relación para $\Delta\eta_k$ en el método de Davidon - Fletcher - Powell, $y_k = \Delta x_k$ y $Z_k = \eta_k \Delta g_k$ con lo que al substituir en (4) se obtiene

$$\eta_{k-1} = \eta_k + A_L + B_k$$

$$= \eta_k + \frac{\Delta x_k (\Delta x_k)^T}{(\Delta x_k)^T \Delta x_k} - \frac{\eta_k \Delta g_k (\Delta g_k)^T \eta_k^T}{(\Delta g_k)^T \eta_k \Delta g_k} \qquad (5)$$

en donde las matrices A_k y B_k son simétricas y si, además, η_k es

también simétrica, entonces η_{k+1} también lo será. La relación anterior (Ec. (5)) produce resultados satisfactorios en la práctica siempre y cuando

1. El error al evaluar $\nabla f(x_k)$ no sea grande

2. η_k no se haga mal-condicionada

El papel de la matriz A_k en la ecuación (5) es garantizar que $\eta \longrightarrow H^{-1}$, mientras que la matriz B_k garantiza que η_{k+1} sea positiva definida en todos los pasos, y en el limite se cancela con η_o. Esto se puede ver como sigue

$$\eta_1 = I + A_o - B_o$$

$$\eta_2 = \eta_1 + A_1 - B_1 = I + (A_o + A_1) - (B_o - B_1)$$

$$\eta_{k+1} = I + \sum_{i=0}^{k} A_k - \sum_{i=0}^{k} B_k$$

Para una función cuadrática la suma de las matrices A_i debe ser igual a H^{-1} cuando $k = n - 1$, y la suma de las matrices B_i deberá cancelar η_o (I en este caso), se puede decir que el método de Davidon - Fletcher - Powell refleja, en cierta forma, toda la información ganada en iteraciones anteriores, a través de η.

Debe señalarse que el método que se está descubriendo usa direcciones conjugadas si la función objetivo es cuadrática. Para que la última dirección, S_{n-1}, sea conjugada a todas las anteriores, se debe

cumplir que :

$$X^{T}_{n-1} H \, s_{n-1} = 0$$

si se substituye que $S_{n-1} = - \eta_{n-1} \nabla f(x_{n-1})$, entonces

$$X^{T}_{n-1} H_{n-1} f(x_{n-1}) = 0 \qquad (6)$$

donde $X_{n-1} = \left(\Delta x_0, \Delta x_1, \ldots, \Delta x_{n-1} \right)$. Si $H \eta_{n-1} = I$ ($\eta_{n-1} = H^{-1}$), entonces $\nabla f(x_{n-1})$ es conjugada a todas las direcciones de búsqueda anteriores dadas por $\Delta x_0, \Delta x_1, \ldots, \Delta x_{n-1}$. Sabiendo que todas las direcciones de búsqueda son conjugadas, se puede demostrar que $\sum_{i=0}^{n-1} A_i = H^{-1}$, como sigue. Como $\Delta g_k = H \cdot \Delta x_k$, entonces el numador y denominador de cada A_i es

$$(\Delta x_k)(\Delta x_k)^T = (\alpha_k S_k)(\alpha_k S_k)^T = \alpha_k^2 S_k S_k^T$$

$$(x_k)^T g_k = (\alpha_k S_k)^T (H \alpha_k S_k) = \alpha_k^2 S_k^T H S_k$$

de donde

$$\sum_{i=0}^{n-1} A_i = \sum_{i=0}^{n-1} \frac{S_i S_i^T}{S_i^T H S_i} \qquad (7)$$

que es la fórmula (4 b). Sec. 2.3, obtenida con anterioridad.

Para terminar la presentación de este método, se hacen dos comentarios sobre la implementación práctica del mismo

1. En algunos problemas, los métodos de métrica variable fallan en alcanzar el mínimo de la función objetivo si el grado de precisión en la búsqueda unidimensional no es suficientemente . Se recomienda que la precisión en la búsqueda unidireccional sea al menos equivalente que en que se requiere para detener al algoritmo completo.

2. La búsqueda por el mínimo se debe detener, si al evaluar los vectores $-\eta_k \Delta f(x_k)$ y $-\alpha_k \eta_k \nabla f(x_k)$ ocurre cualquiera de los dos siguientes puntos.

   a) Cada componente en ambos rectores es menor que una tolerancia dada.

   b) Las longitudes predichas al mínimo, de cualquiera de los dos vectores es inferior a una cierta tolerancia.

2.4.3. Algoritmos de Pearson

Pearson propusó una serie de algoritmos para calcular η_1 usando direcciones que fueran conjugadas. Los algoritmos de Pearson se pueden obtener empleando diferentes vectores y, Z en la ecuación (4) del inciso 2.4., según se muestra a continuación

1. Pearson No. 2 Sea $y = Z = \Delta x_k$ y $w = 1$.

Entonces

$$\eta_{k+1} = \eta_k + \frac{(\Delta x_k - \eta_k \Delta g_k)(\Delta x_k)^T}{(\Delta x_k)^T \Delta g_k}$$

$$\eta_o = R_o$$

donde R_o es cualquier simétrica positiva definida. Este algoritmo generalmente conduce a matrices mal condicionadas.

2. Pearson No. 3 Sea $y = Z = \eta_k \Delta g_k$, con $w = 1$.

Entonces

$$\eta_{k+1} = \eta_k + (\Delta x_k - \eta_k \Delta g_k) \frac{(\eta_k \Delta g_k)^T}{(\Delta g_k)^T \eta_k \Delta g_k}$$

$$\eta_o = R_o$$

Este algoritmo se comporta bastante parecido al de Davidon - Fletcher - Powell, excepto que el tamaño del paso es, en general, inferior al de este último.

3. Newton - Raphson Proyectado.

Pearson propuso este otro algoritmo al cual se puede obtener haciendo que $x \longrightarrow \infty$ y $Z = \eta_k \Delta g_k$, con lo que se obtiene.

$$\eta_{k+1} = \eta_k - \frac{(\eta_k \Delta g_k)(\eta_k \Delta g_k)^T}{(\Delta g_k)^T \eta_k \Delta g_k}$$

$$\eta_o = R_o$$

Este método incluye la siguiente regla de reinicio cada n etapas, donde n es el número de variables independientes, sobre la matriz η_k.

$$R_{k-1} = R_k + \frac{(\Delta x_k - R_k \Delta g_k)(\eta_k \Delta g_k)^T}{(\Delta g_k)^T \eta_k \Delta g_k}$$

es decir, cada n etapas se toma $\eta_k = R_k$.

B I B L I O G R A F I A

1. C.G. Broyden, Math. Computation, 21 : 368 (1967)

2. M.J. Box, D. Davis, and W.H. Swann. "Non-linear Optimization Techniques", Chemical Industries Monograph 5, Oliver and Boyd, Edinburgh, 1970

3. R.G. Coggins, Univariate Search Methods, Imperial Chemical Industries Ltd., Central Instr. Lab. Res. Note 64/11, 1964

4. W.C. Davidon, USAEC Doc. ANL - 5990 (Rev.), Nov., 1959

5. W.C. Davidon, Computes J., 10 : 406 (1908) ; Chap. 2 in R. Fletcher (ed.), "Optimization" Academic Press Inc. N.Y., 1969

6. R. Fletcher and M.J.D. Powell, Computer J., 6 : 163 (1963)

7. R. Fletcher and C.M. Reeves, Computer J., 7 : 149 (1964)

8. M.R. Hestenes, The Conjugate Gradient Method for Solving Linear Systems, in Proceedings of the Symposium on Applied Mathematics, Vol. VI, Mc. Graw-Hill Book Company, New York, 1956, pp. 83-102

9. M.R. Hestenes and E.L. Steifel, J. Res. Nat. Bur. Std., B 49 : 409 (1952)

10. J.D. Pearson, Computer J., 13 : 171 (1969)

11. M.J.D. Powell, Computer J., 7 : 155 (1964)

12. M.J.D. Powell, Rank One Methods for Unconstrained Minimization, AERE Rept., TP 372, 1969

13. D.J. Wilde, "Optimum Seeking Methods", Prentice Hall, Inc., Englewood Cliffs, N.J., 1964.

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

DISEÑO OPTIMO DE SISTEMAS DE INGENIERIA

~~OPTIMACION DE SISTEMAS FISICOS~~

METODOS DE OPTIMACION CON RESTRICCIONES

DR. ALEJANDRO VELASCO LEVY

MARZO, 1981

AERO-ASTRONAUTICS REPORT NO. 55

GRADIENT METHODS IN MATHEMATICAL PROGRAMMING PART I - REVIEW OF PREVIOUS TECHNIQUES

by

A. MIELE, H.Y. HUANG, and J.W. CANTRELL

RICE UNIVERSITY

1969

Gradient Methods in Mathematical Programming

Part I - Review of Previous Techniques[1]

by

A. MIELE[2], H.Y. HUANG[3], AND J.W. CANTRELL[4]

Abstract. This report is the first of a series on gradient methods in mathematical programming. It considers the problem of minimizing a function $f(x)$, where f is a scalar function and x is an n-vector whose components are unconstrained. For this problem, three previous methods are reviewed, namely, the ordinary gradient method, the conjugate-gradient method, and the variable-metric method. A new intuitive derivation of the last two algorithms is presented.

[1] This research was supported by the Office of Scientific Research, Office of Aerospace Research, United States Air Force, Grant No. AF-AFOSR-828-67.

[2] Professor of Astronautics and Director of the Aero-Astronautics Group, Department of Mechanical and Aerospace Engineering and Materials Science, Rice University, Houston, Texas.

[3] Assistant Professor of Astronautics, Department of Mechanical and Aerospace Engineering and Materials Science, Rice University, Houston, Texas.

[4] Graduate Student in Aero-Astronautics, Department of Mechanical and Aerospace Engineering and Materials Science, Rice University, Houston, Texas.

1. Definitions

The following definitions are used throughout the paper:

(a) The symbol x denotes the position vector

$$x = \begin{bmatrix} x^1 \\ x^2 \\ \cdot \\ \cdot \\ \cdot \\ x^n \end{bmatrix} \tag{1}$$

whose scalar components are $x^1, x^2, \ldots, x^n$.

(b) The symbol f denotes a scalar function of the vector x, that is

$$f = f(x) \tag{2}$$

(c) The symbol g denotes the column vector

$$g(x) = \begin{bmatrix} \partial f/\partial x^1 \\ \partial f/\partial x^2 \\ \cdot \\ \cdot \\ \cdot \\ \partial f/\partial x^n \end{bmatrix} \tag{3}$$

whose components are the first partial derivatives of f with respect to the scalar variables $x^1, x^2, \ldots, x^n$. This is the gradient of the function f.

(d) The symbol H denotes the square matrix

$$H(x) = \begin{bmatrix} \partial^2 f/\partial x^1 \partial x^1 & \partial^2 f/\partial x^1 \partial x^2 & \ldots & \partial^2 f/\partial x^1 \partial x^n \\ \partial^2 f/\partial x^2 \partial x^1 & \partial^2 f/\partial x^2 \partial x^2 & \ldots & \partial^2 f/\partial x^2 \partial x^n \\ \cdot & \cdot & \cdot & \cdot \\ \partial^2 f/\partial x^n \partial x^1 & \partial^2 f/\partial x^n \partial x^2 & \ldots & \partial^2 f/\partial x^n \partial x^n \end{bmatrix} \tag{4}$$

whose components are the second partial derivatives of the function f with respect to the scalar variables $x^1, x^2, \ldots, x^n$.

(e) The symbol x denotes the nominal point. The symbol $\bar{x}$ denotes the point following x. The symbol $\underline{x}$ denotes the point preceding x.

(f) The symbol $\delta(\ldots)$ denotes the displacement leading from a point to the next point. Therefore, the following relations hold:

$$\begin{aligned} \bar{x} &= x + \delta x \\ x &= \underline{x} + \delta \underline{x} \end{aligned} \tag{5}$$

(g) The superscript T denotes the transpose of a matrix.

2. Introduction

A basic problem of mathematical programming is that of finding the minimum of a function

$$f = f(x) \tag{6}$$

where f is a scalar function and x is an n-vector. If the n components of the vector x are unconstrained, the extremum of (6) occurs when the following necessary condition is satisfied:

$$g(x) = 0 \tag{7}$$

where g is the gradient of the function f with respect to the vector x. For a minimum, the matrix of the second derivatives (4) must be positive definite at the point x defined by (7).

If the function (6) is quadratic, the gradient g(x) is linear with respect to x. Hence, Eq. (7) can be solved analytically. On the other hand, if (6) is nonquadratic, the gradient g(x) is nonlinear. This being the case, approximate methods must be employed to solve Eq. (7). One possible method consists of quasilinearizing (7) about a nominal point. Another method, the descent method, consists of constructing corrections for leading from a nominal point x to a varied point $\bar{x}$ such that

$$f(\bar{x}) < f(x) \tag{8}$$

Thus, by an iterative procedure (that is, through successive decreases in the value of the function f), it is hoped that the minimum of f is approached to any desired degree of accuracy.

This report is the first of a series on gradient methods in mathematical programming. It reviews three of the existing techniques, namely, the ordinary gradient method, the conjugate-gradient method (Refs. 1-3), and the variable-metric method (Refs. 4-5). For the last two methods, a new intuitive derivation is presented.

3. Ordinary Gradient Method

To first-order terms, the values of the function (6) at the varied point and the central point are related by

$$f(\bar{x}) = f(x) + \delta f(x) \tag{9}$$

where the first variation $\delta f(x)$ is given by

$$\delta f(x) = g^T(x)\delta x \tag{10}$$

$$= \delta x^T g(x)$$

$$\delta x = \xi - x \tag{11}$$

Also to first-order terms, the greatest decrease in the value of the function is achieved if the first variation (10) is minimized. Here, we limit our analysis to those variations δx which satisfy the constraint

$$K = \delta x^T \delta x \tag{12}$$

where K is a prescribed quantity.

3.1. Derivation of the Algorithm. Standard methods of the theory of maxima and minima show that the fundamental function of this problem is the scalar function

$$F = g^T(x)\delta x + (1/2\alpha)\delta x^T \delta x \tag{13}$$

where $1/2\alpha$ is a constant Lagrange multiplier. The optimum system of variations must be such that

$$G(\delta x) = 0 \tag{14}$$

where G is the gradient of the function F with respect to the scalar variables $\delta x^1, \delta x^2, \ldots, \delta x^n$. In the light of (13), the explicit form of (14) is the following:

$$\delta x = -\alpha g(x) \tag{15}$$

and shows that the optimum correction δx has the gradient direction. This is why the method is called the ordinary gradient method. Upon substituting (15) into (12), we see that

$$K = \alpha^2 g^T(x) g(x) \tag{16}$$

Therefore, a one-to-one correspondence exists between the value of the constant K and the value of α. This being the case, one can bypass prescribing K and reason directly on α, as in the considerations which follow.

3.2. Descent Property. Upon combining Eqs. (10) and (15), we see that the first variation becomes

$$\delta f(x) = -\alpha g^T(x) g(x) \tag{17}$$

and is negative for $\alpha > 0$. Therefore, if α is sufficiently small, the function f decreases. This guaranteed decrease of f at every step is the most important property of the gradient method.

3.3 Optimum Stepsize. The next step is to assign a value to the parameter α, the stepsize of the gradient method. There are two situations to be avoided: (a) if the stepsize α is excessivelingly small, the decrease of the function is guaranteed but very small; therefore, the number of iterations necessary for convergence is large; and (b) if the stepsize α is too large, the first variation may be only a small part of the total variation; therefore, the function f may actually increase. In order to prevent the occurrence of (a) and (b), the stepsize α must be in the proper range.

If Eqs. (11) and (15) are combined, the position vector at the end of any iteration becomes

$$\bar{x} = x - \alpha g(x) \tag{18}$$

For each point x, Eq. (18) defines a one-parameter family of points $\bar{x}$ for which the function f takes the form

$$f(\bar{x}) = f(x - \alpha g(x)) = F(\alpha) \tag{19}$$

The greatest decrease in the function $F(\alpha)$ occurs if the parameter α satisfies the following necessary condition:

$$F_\alpha = 0 \tag{20}$$

On account of (19), the following relation holds:[5]

$$F_\alpha = -g^T(\bar{x})g(x) \tag{21}$$

[5] At $\alpha = 0$, the derivative (21) is given by $F_\alpha = -g^T(x)g(x)$ and is negative, since $g^T(x)g(x) > 0$.

Therefore, Eq. (20) becomes

$$g^T(\bar{x})g(x) = 0$$

and shows that the gradient $g(\bar{x})$ is orthogonal to the gradient $g(x)$.

3.4. Summary of the Algorithm. Equations (15), (17), and (22) summarize the general properties of the ordinary gradient algorithm. They are valid regardless of the function $f(x)$, as long as it is continuous and has continuous first derivatives.

For any given iteration, the algorithm can be summarized as follows: (a) for a given nominal point x, the gradient $g(x)$ is known from Eq. (3); (b) the optimum value of the stepsize α must be determined by solving Eq. (20) with a one-dimensional search, as in Section 7; (c) the correction δx to the position vector x is determined using Eq. (15); and (d) the new position vector $\bar{x}$ is computed through Eq. (11). Next, the position vector $\bar{x}$ becomes the nominal point x for the subsequent iteration, and the procedure is repeated until a predetermined stopping condition is satisfied (see Section 8).

4. Modification of the Ordinary Gradient Method

The ordinary gradient method is conceptually simple and stable in that the function $f(x)$ is reduced at every iteration; however, it has the drawback of slow convergence. For this reason, methods have been developed to reduce the number of iterations required for convergence. In this connection, let the displacement vector δx be written in the form

$$\delta x = -\alpha p \tag{23}$$

where p is the search direction. The following are particular forms of the vector p:

$$p = g(x) + q \tag{24}$$

and

$$p = Ag(x) \tag{25}$$

where q is an n-vector and A is an $n \times n$ symmetric matrix. In Section 5, the conjugate-gradient algorithm is derived by reasoning on (24); in Section 6, the variable-metric algorithm is derived by reasoning on (25).

5. Conjugate-Gradient Method

In this section, we consider the algorithm

$$\bar{x} = x + \delta x, \qquad \delta x = -\alpha p, \qquad p = g(x) + \eta \tag{26}$$

where η is an n-vector to be specified. The first variation of the function (6) is given by Eq. (10) which, in the light of (26), becomes

$$\delta f(x) = -\alpha[g^T(x)g(x) + g^T(x)\eta] \tag{27}$$

We note that $g^T(x)g(x) > 0$. Therefore, for $\alpha > 0$, the descent property of this algorithm is ensured if one chooses η so that

$$g^T(x)\eta = 0 \tag{28}$$

If Eqs. (26-1) and (26-2) are combined, the position vector at the end of any i iteration becomes

$$\bar{x} = x - \alpha p \tag{29}$$

For a given point x and a given vector p, Eq. (29) defines a one-parameter family of points $\bar{x}$ for which the function f takes the form

$$f(\bar{x}) = f(x - \alpha p) = F(\alpha) \tag{30}$$

The greatest decrease in the function $F(\alpha)$ occurs if the parameter α satisfies the following necessary condition:

$$F_\alpha = 0 \tag{31}$$

On account of (), the following relation holds:

$$F_\alpha = -g^T(\bar{x})p \tag{32}$$

Therefore, Eq. (31) becomes

$$g^T(\bar{x})p = 0 \tag{33}$$

and shows that the gradient $g(\bar{x})$ is orthogonal to the search direction p.

Next, we apply Eq. (33) to the previous iteration and obtain

$$g^T(x)\beta = 0 \tag{34}$$

By comparing (28) and (34), we conclude that one possible choice of the vector q is the following:

$$q = \gamma\beta \tag{35}$$

where γ is a constant. As a consequence, the algorithm (28) can be rewritten as

$$\bar{x} = x + \delta x\,, \quad \delta x = -\alpha p\,, \quad p = g(x) + \gamma\beta \tag{36}$$

The next step is to determine the constant γ. If Eqs. (36) are combined, the position vector at the end of any iteration becomes

$$\bar{x} = x - \alpha g(x) + \alpha\gamma\beta \tag{37}$$

For a given point x and a given vector β, Eq. (37) defines a two-parameter family of points $\bar{x}$ for which the function f takes the form

$$f(\bar{x}) = f(x - \alpha g(x) + \alpha\gamma\beta) = F(\alpha, \gamma) \tag{38}$$

The greatest decrease in the function $F(\alpha, \gamma)$ occurs if the parameters α and γ satisfy the following necessary conditions:

$$F_\alpha = 0\,, \quad F_\gamma = 0$$

On account of (36-3) and (38), the following relations hold:

$$F_\alpha = -g^T(\bar{x})p\,, \quad F_\gamma = -\alpha g^T(\bar{x})\beta \tag{39}$$

Therefore, Eqs. (39) become

$$g^T(\bar{x})p = 0\,, \quad g^T(\bar{x})\beta = 0 \tag{41}$$

and show that the gradient $g(\bar{x})$ is orthogonal to the search directions p and β. A mathematical consequence of Eqs. (36-3) and (41) is that

$$g^T(\bar{x})g(x) = 0$$

showing that the gradients $g(\bar{x})$ and g(x) are orthogonal.

3.1. Quadratic Function. Now, consider the particular case of a quadratic function, that is, a function of the form

$$f(x) = a + b^T x + \frac{1}{2}x^T H x \tag{43}$$

where a is a constant scalar, b is a constant n-vector, and H is a constant, symmetric n x n matrix. For this function, the gradient is a linear function of x, that is,

$$g(x) = b + Hx \tag{44}$$

Since

$$g(\bar{x}) = b + H\bar{x} \tag{45}$$

relations (14) and (45) imply that

$$g(\bar{x}) = g(x) + H\delta x = g(x) - \alpha H p \tag{46}$$

Next, we introduce Eqs. (46) into (41) and, after laborious manipulations, obtain the solutions (Ref. 1)

$$\alpha = \frac{g^T(x)p(x)}{p^T H p} \qquad \gamma = \frac{g^T(x)g(x)}{g^T(\bar{x})g(\bar{x})} \tag{47}$$

where p is given by (16-3).

For a quadratic function, Hestenes and Stiefel (Ref. 1) proved that, if the first step of the descent process is a gradient step, the following relations hold:

$$g^T(x)g(x_i) = 0 \quad , \quad g^T(x)p_i = 0 \quad , \quad p^T H p_i = 0 \tag{48}$$

where x_i denotes any state preceding x. Equation (48-1) states that the gradient at each iteration is orthogonal to the gradient at every previous iteration. Equation (48-2) states that the gradient at each iteration is orthogonal to the search direction at every previous iteration. Finally, Eq. (48-3) states that the search direction at each iteration and the search direction at every previous iteration are conjugate with respect to the constant matrix H; this is why the algorithm is called the conjugate-gradient method. The algorithm (16) with α and γ defined by (17) reduces the gradient to zero in no more than n steps; therefore, the minimum of f(x) is reached in no more than n steps.

5.2. Nonquadratic function. For a nonquadratic function, solving Eqs. (41) for α and γ requires a two-dimensional search (Ref. 6). The difficulty of this process can be avoided if one optimizes α exactly and uses an approximate value for γ, namely, that given by Eq. (47-2). This leads to the algorithm (Ref. 3)

$$\bar{x} = x + \delta x, \quad \delta x = -\alpha p \quad , \quad p = g(x) + \frac{g^T(x)g(x)}{g^T(\bar{x})g(\bar{x})}\beta \tag{49}$$

in which α is optimized by searching for the minimum of f along the direction defined by (49). Theoretically, therefore, the optimization of α requires that the relation (41-1) be satisfied.

For any iteration except the first, the complete algorithm can be stated as follows: (a) for a given nominal point x, the gradient g(x) is known; since the gradient $g(\bar{x})$ and the search direction β are known from the previous iteration, the search direction p can be determined with Eq. (49-3); (b) the optimum stepsize α must be determined by minimizing the function f along the search direction p, as in Section 7; (c) the correction δx to the position vector x is determined using Eq. (49-2); and (d) the new position vector $\bar{x}$ is computed through Eq. (49-1). Next, the position vector $\bar{x}$ becomes the nominal point for the subsequent iteration, and the procedure is repeated until a predetermined stopping condition is satisfied (see Section 8). To start the algorithm, one bypasses (49-3) and sets p = g(x), equivalent to stating that the first step is a gradient step.

In closing, the following comments are pertinent: (a) in the conjugate-gradient method, it is important that the stepsize α be determined accurately, while this is not the case with the ordinary gradient method; (b) theoretical considerations and numerical experience show the desirability of restarting the process every n or n + 1 iterations, that is, resetting p = g(x) every n or n + 1 iterations (Ref. 3).

6. Variable Metric Algorithm

In this section, we consider the algorithm

$$\bar{x} = x + \delta x, \qquad \delta x = -\alpha p, \qquad p = A g(x) \tag{50}$$

where A is a symmetric n x n matrix to be specified. The first variation of the function (6) is given by Eq. (10) which, in the light of (50), becomes

$$\delta f(x) = -\alpha g^T(x) A g(x) \tag{51}$$

We note that, if the matrix A is positive definite, $g^T(x)Ag(x) > 0$. Therefore, for $\alpha > 0$, the descent property of this algorithm is ensured.

Now, consider the points $\hat{x}$ and x. At point $\hat{x}$, the gradient $g(\hat{x})$ and the matrix $\hat{A}$ are known; at point x, the gradient g(x) is known. Therefore, the differences

$$\delta x = x - \hat{x}, \qquad \delta g = g(x) - g(\hat{x}) \tag{52}$$

are available. We wish to determine the matrix A so that the relation

$$A \delta g = \delta x \tag{53}$$

is satisfied. After defining the matrix difference

$$\delta A = A - \hat{A} \tag{54}$$

we combine (53)-(54) to obtain

$$\delta A \, \delta g = \delta x - \hat{A} \delta g \tag{55}$$

in which the only unknown is δA. Equation (55) admits the solution

$$\delta A = \frac{\delta x y^T}{y^T \delta g} - \frac{\hat{A} \delta g z^T}{z^T \delta g} \tag{56}$$

where y and z denote arbitrary n-vectors. Therefore, the matrix A must be updated according to the relation

$$A = \hat{A} + \frac{\delta x y^T}{y^T \delta g} - \frac{\hat{A} \delta g z^T}{z^T \delta g} \tag{57}$$

In particular, if one chooses

$$y = \delta x, \qquad z = \hat{A} \delta g \tag{58}$$

Eq. (57) becomes

$$A = \hat{A} + \frac{\delta x \delta x^T}{\delta x^T \delta g} - \frac{\hat{A} \delta g \delta g^T \hat{A}}{\delta g^T \hat{A} \delta g} \tag{59}$$

Note that the second and third matrices on the right-hand side of (59) are symmetric; therefore, if $\hat{A}$ is symmetric, A is also symmetric.

6.1. Quadratic Function. Now, consider the particular case of a function having the form (43). For this quadratic function, the following properties can be shown to hold (Refs. 4-5 and 7-8):

(a) If the initial matrix A is chosen to be the inverse of the second derivative matrix H, that is, if

$$A = H^{-1}$$

at the initial point, the variable-metric algorithm exhibits one-step convergence. This is due to the fact that the variable-metric algorithm becomes identical with quasilinearization.

(b) If the initial matrix A is chosen to be positive definite, any subsequent matrix A is also positive definite. With this understanding, the following relations hold:

$$g^T(x_{i+1})p_i = 0, \quad p^T H p_i = 0 \tag{61}$$

Equation (61-1) states that the gradient at each iteration is orthogonal to the search direction at every previous iteration. Equation (61-2) states that the search direction at each iteration and the search direction at every previous iteration are conjugate with respect to the constant matrix H. As the algorithm progresses, the matrix A tends to the inverse of the second derivative matrix H, and relation (60) becomes satisfied exactly when convergence is achieved. The algorithm (50), with A updated according to (59), reduces the gradient to zero in no more than n steps; therefore, the minimum of f(x) is reached in no more than n steps.

(c) As a particular case of (b), the initial matrix can be chosen to be

$$A = I \tag{62}$$

where I is the n x n identity matrix. Under these conditions, the variable-metric algorithm becomes identical with the conjugate-gradient algorithm of Section 5.

6.2. Nonquadratic Function. For a nonquadratic function, the variable-metric algorithm is represented by

$$\bar{x} = x + \lambda x, \quad \lambda x = -\alpha p, \quad p = Ag(x) \tag{63}$$

with

$$A = \hat{A} + \frac{\alpha\delta\bar{x}^T}{\delta\bar{x}^T\delta\bar{g}} - \frac{\hat{A}\delta\bar{g}\delta\bar{g}^T\hat{A}}{\delta\bar{g}^T\hat{A}\delta\bar{g}} \tag{64}$$

The stepsize α is to be optimized by searching for the minimum of f along the direction defined by (63).

For any iteration except the first, the complete algorithm can be stated as follows: (a) for a given nominal point x, the gradient g(x) is known; since $g(\bar{x})$, $\delta\bar{x}$, $\hat{A}$ are known from the previous iteration, the matrix A can be computed with (64) and the search direction p with (63-2); (b) the optimum stepsize α must be determined by minimizing the function f along the search direction p, as in Section 7; (c) the correction δx to the position vector x is determined using Eq. (63-2); and (d) the new position vector $\bar{x}$ is computed through Eq. (63-1). Next, the new position vector $\bar{x}$ becomes the nominal point for the subsequent iteration and the procedure is repeated until a predetermined stopping condition is satisfied (see Section 8). To start the algorithm, one bypasses (64) and sets A equal to any symmetric, positive-definite matrix (for instance, the identity matrix).

In closing, the following comments are pertinent: (a) in the variable-metric method, it is important that the stepsize α be determined accurately; (b) restarting the algorithm every n or n + 1 iterations is not necessary; however, restarting is indispensable whenever the positive-definiteness of the matrix A is violated, for example, if the stepsize α becomes negative.

7. Search Techniques

In each of the previous methods, the stepsize α must be optimized. In this section, we present techniques to solve the equation

$$F_\alpha(\alpha) = 0 \tag{65}$$

(that is, to find the minimum of the function $F(\alpha)$ given by Eq. (19) for the ordinary gradient algorithm or Eq. (30) for the conjugate-gradient and variable-metric algorithms. Since the techniques in question involve the consideration of the first derivative F_α and perhaps the second derivative $F_{\alpha\alpha}$, we summarize these derivatives below.

For all of the previous methods, we have

$$F_\alpha(\alpha) = -g^T(\bar{x})p\,, \quad F_{\alpha\alpha}(\alpha) = p^T H(\bar{x})p \tag{66}$$

where

$$\bar{x} = x + \alpha p \tag{67}$$

The search direction is given by $p = g(x)$ for the ordinary gradient method, Eq. (49-3) for the conjugate-gradient method, and Eq. (63-3) for the variable-metric method. Of course, Eq. (66-2) requires that the second-derivative matrix $H(x)$ be explicitly available. If this is not the case, one can use the difference scheme

$$F_{\alpha\alpha}(\alpha) = (1/2\theta)[F_\alpha(\alpha + \theta) - F_\alpha(\alpha - \theta)] \tag{68}$$
$$= (1/2\theta)[g(\bar{x} + \theta p) - g(\bar{x} - \theta p)]^T p$$

In practice, one may choose

$$\theta = \epsilon_1/|p| \tag{69}$$

where ϵ_1 is a small number.

7.1. Cubic Interpolation. Let the values of the function $F(\alpha)$ and the derivative $F_\alpha(\alpha)$ be computed for two different values of α, namely, α_1 and α_2, with $\alpha_2 > \alpha_1 > 0$. If α_1 and α_2 are such that

$$F_\alpha(\alpha_1) < 0\,, \quad F_\alpha(\alpha_2) > 0$$

then the minimum of the function $F(\alpha)$ occurs for some value α in the range

$$\alpha_1 < \alpha < \alpha_2 \tag{71}$$

In this range, we represent the function $F(\alpha)$ with the cubic

$$F(\alpha) = A + B(\alpha - \alpha_1) + C(\alpha - \alpha_1)^2 + D(\alpha - \alpha_1)^3 \tag{72}$$

whose first and second derivatives are given by

$$F_\alpha(\alpha) = B + 2C(\alpha - \alpha_1) + 3D(\alpha - \alpha_1)^2\,, \quad F_{\alpha\alpha}(\alpha) = 2C + 6D(\alpha - \alpha_1) \tag{73}$$

The scalar coefficients A, B, C, D are determined by requiring (72) to match the ordinate and the slope of the curve $F(\alpha)$ at α_1 and α_2. Therefore, one has to solve the linear equations

$$F(\alpha_1) = A\,, \quad F(\alpha_2) = A + B(\alpha_2 - \alpha_1) + C(\alpha_2 - \alpha_1)^2 + D(\alpha_2 - \alpha_1)^3$$
$$F_\alpha(\alpha_1) = B\,, \quad F_\alpha(\alpha_2) = B + 2C(\alpha_2 - \alpha_1) + 3D(\alpha_2 - \alpha_1)^2 \tag{74}$$

Once the coefficients of the cubic (72) are known, the optimum value of α is determined by the condition (65). Therefore, in the light of (73), one arrives at the solution

$$\alpha = \alpha_1 + (1/3D)[-C + \sqrt{(C^2 - 3BD)}] \tag{75}$$

At this point, we recompute the function $F(\alpha)$ and the derivative $F_\alpha(\alpha)$. Then, the process is iterated until a predetermined stopping condition is satisfied. For instance, one may require that

$$|F_\alpha(\alpha)| \leq \epsilon_2 \tag{76}$$

or that

$$|F_\alpha(\alpha)| \leq \epsilon_3 |F_\alpha(0)| \tag{77}$$

where ϵ_2 and ϵ_3 are prescribed small numbers.

7.2. Quasilinearization. An alternate technique for computing the optimum stepsize, that of quasilinearization with built-in safeguards to ensure that the function decreases at every step of the iterative search, is now presented. Let

$$\delta\alpha = \alpha - \alpha_0 \tag{78}$$

denote the correction to α starting from an arbitrary nominal value α_0. If quasilinearization is applied to Eq. (65), one obtains the linear algebraic equation

$$F_{\alpha\alpha}(\alpha_0)\delta\alpha + F_\alpha(\alpha_0) = 0 \tag{79}$$

Next, we imbed Eq. (79) in the more general equation

$$F_{\alpha\alpha}(\alpha_0)\delta\alpha + \mu\rho F_\alpha(\alpha_0) = 0 \tag{80}$$

where μ denotes a scaling factor and ρ a direction factor such that

$$0 < \mu \leq 1 \ , \quad \rho = \pm 1 \tag{81}$$

Equation (80) admits the solution

$$\delta\alpha = -\mu\rho F_\alpha(\alpha_0)/F_{\alpha\alpha}(\alpha_0) \tag{82}$$

The direction factor ρ is determined in such a way that the first variation

$$\delta F(\alpha_0) = F_\alpha(\alpha_0)\delta\alpha \tag{83}$$

is negative. From (82)-(83), we obtain

$$\delta F(\alpha_0) = -\mu\rho F_\alpha^2(\alpha_0)/F_{\alpha\alpha}(\alpha_0) \tag{84}$$

Therefore, $\delta F(\alpha_0)$ is negative if the direction factor ρ is chosen as follows:

$$\rho = \text{sign } F_{\alpha\alpha}(\alpha_0) \tag{85}$$

Because of this choice, the correction (82) becomes

$$\delta\alpha = -\mu F_\alpha(\alpha_0)/|F_{\alpha\alpha}(\alpha_0)| \tag{86}$$

To perform the search, a nominal value must be given to α_0. Then, one sets $\mu = 1$, computes $\delta\alpha$ from Eq. (86) and α from Eq. (78). If $F(\alpha) < F(\alpha_0)$, the scaling factor $\mu = 1$ is acceptable. If $F(\alpha) > F(\alpha_0)$, the previous value of μ must be replaced by some smaller value in the range $0 < \mu < 1$ until the condition $F(\alpha) < F(\alpha_0)$ is met; this can be obtained through bisection, that is, by successively dividing the value of μ by 2. At this point, the search step is completed. The value obtained for α becomes the nominal value α_0 for the next search step, and the procedure is repeated until a desired degree of accuracy is obtained, that is, until Ineq. (76) or (77) is satisfied. In the absence of better information, the first step of the search procedure can be made with $\alpha_0 = 0$.

8. Termination of the Algorithm

One way to terminate the gradient algorithm is to impose a condition on the modulus of the gradient, for example,

$$g^T(x)g(x) \leq \epsilon_4 \tag{87}$$

where ϵ_4 is a prescribed small number. If the function f(x) is rather flat in the neighborhood of the minimum, then Eq. (87) may not yield precise coordinates. In this case, the following additional condition is suggested:

$$\Delta x^T \Delta x < \epsilon_5 \tag{88}$$

where ϵ_5 is a prescribed small number.

References

1. HESTENES, M.R., and STIEFEL, E., Methods of Conjugate Gradients for Solving Linear Systems, Journal of Research of the National Bureau of Standards, Vol. 49, No. 6, 1952.
2. BECKMAN, F.S., The Solution of Linear Equations by the Conjugate Gradient Method, Mathematical Methods for Digital Computers, Vol. 1, Edited by A. Ralston and H.S. Wilf, John Wiley and Sons, New York, 1960.
3. FLETCHER, R., and REEVES, C.M., Function Minimization by Conjugate Gradients, Computer Journal, Vol. 7, No. 2, 1964.
4. DAVIDON, W.C., Variable-Metric Method for Minimization, Argonne National Laboratory, Report No. ANL-5990, 1959.
5. FLETCHER, R., and POWELL, M.J.D., A Rapidly Convergent Descent Method for Minimization, Computer Journal, Vol. 6, No. 2, 1963.
6. MIELE, A., and CANTRELL, J.W., Gradient Methods in Mathematical Program Part 2, Memory Gradient Method, Rice University, Aero-Astronautics Report No. 56, 1969.
7. MYERS, G.E., Properties of the Conjugate Gradient and Davidon Methods, Journal of Optimization Theory and Applications, Vol. 2, No. 4, 1968.
8. PEARSON, J.D., On Variable Metric Methods of Minimization, Research Analysis Corporation, Technical Paper No. RAC-TP-302, 1968.

RICERCHE DI AUTOMATICA
Vol. 1, No. 1, August 1970
Printed in Italy

UPDATING RULES FOR THE PENALTY CONSTANT USED IN THE PENALTY FUNCTION METHOD FOR MATHEMATICAL PROGRAMMING PROBLEMS

A. MIELE, G. M. COGGINS, and A. V. LEVY

Abstract. The problem of minimizing a function $f(x)$ subject to a constraint $\varphi(x)=0$ is considered, where f is a scalar, x an n vector, and φ a q vector, with $q \leq n$. The penalty function method is investigated; that is, a sequence of unconstrained minimization problems is solved, each of which involves the penalty function $U(x,k)=f(x)+kP(x)$. Here, the penalty constant k is a positive, scalar quantity, and $P(x)=\varphi^T(x)\varphi(x)$ is the norm squared of the constraint error.

Crucial to the penalty function method is the prediction of the ratio $\alpha=k_2/k_1$ at which the penalty constant must be increased when shifting from one cycle of the algorithm to the next. Here, k_1 denotes the penalty constant of the present cycle, and k_2 denotes the penalty constant of the next cycle.

In this paper, two variable-rate updating rules are developed: (I) the updating rule $\alpha=\beta[(P_1'/P_2')]$ and (II) the updating rule $\alpha=\beta[(P_1/P_2)]$, where P_1 is the constraint error at the end of the present cycle, P_2 is the constraint error allowed for convergence, and $\beta>1$. Updating rule (I) tends to produce at the end of the next cycle a constraint error P_2 below the geometric mean of the constraint error at the end of the present cycle and that allowed for convergence; updating rule (II) tends to produce at the end of the next cycle a constraint error P_2 below that allowed for convergence.

In order to evaluate these updating rules, six numerical examples are investigated. The first example deals with a quadratic function subject to linear constraints; the remaining examples deal with nonquadratic functions subject to nonlinear constraints. Each example is solved with the conjugate gradient algorithm for five starting values of the

---

Manuscript received May 12, 1972 and in revised form November 17, 1972.

A. Miele is Professor of Astronautics and Mathematical Sciences, Rice University, Houston, Texas 77001. G. M. Coggins is Instructor, Department of Mechanics, US Military Academy, West Point, New York 10996. A. V. Levy is Research Associate, University of Mexico, Mexico City 20, Mexico.

This research, supported by the National Science Foundation, Grant No. GP-32459, is a condensation of the investigation described in Ref. 1.

penalty constant k_1, ranging between 10^{-3} and 10^3. For each example, the variable-rate updating rules (i) and (ii), with $\beta = 10$, are compared with the constant-rate updating rules $\alpha = 5$ and $\alpha = 10$. From the numerical experiments, it is concluded that the variable-rate updating rules are superior to the constant-rate updating rules, in that they generally lead to convergence in a smaller number of iterations.

1. Introduction

Over the past several years, considerable work has been done on the problem of minimizing a function $f(x)$ subject to a constraint $\varphi(x) = 0$ using numerical methods. Here, f is a scalar, x an n-vector, and φ a q-vector, with $q < n$.

The approaches employed are generally based on one of two basic ideas. One is to develop algorithms such that the constraints are satisfied, at least to first order, at the end of each iteration (Refs. 2-4). Another is to develop algorithms involving cycles in which the vector x is viewed as unconstrained and a new function related to $f(x)$ and $\varphi(x)$ is minimized. The penalty function method (Refs. 5-8) is an approach of the latter type.

Crucial in the penalty function method is the prediction of the rate at which the penalty constant must be increased when shifting from one cycle of the algorithm to the next. This key question is considered in this paper, whose objective is the following: to improve the convergence characteristics of the penalty function method by automatically adjusting the penalty constant used in the method.

2. Statement of the Problem

We consider the problem of minimizing the function

$$f = f(x) \tag{1}$$

subject to the constraint

$$\varphi(x) = 0, \tag{2}$$

where f is a scalar, x an n-vector, and φ a q-vector, with $q < n$. Here, all vectors are column vectors. It is assumed that the first and second partial derivatives of the functions $f(x)$ and $\varphi(x)$ exist and are continuous and that the constrained minimum exists.

2.1. *First-Order Conditions.* From theory of maxima and minima, it is known that the above problem is equivalent to that of minimizing the

augmented function

$$F(x, \lambda) = f(x) + \lambda^T \varphi(x) \tag{3}$$

subject to the constraint (2). Here, the q-vector λ is the Lagrange multiplier and the superscript T denotes the transpose of a matrix. If

$$F_x(x, \lambda) = f_x(x) + \varphi_x(x)\,\lambda \tag{4}$$

denotes the gradient of the augmented function, the optimum solution for x and λ must satisfy the relations

$$\varphi(x) = 0, \quad F_x(x, \lambda) = 0, \tag{5}$$

which are a system of $n + q$ equations in the $n + q$ components of x, λ. In Eqs. (4)-(5), the gradients f_x and F_x denote n-vectors and the matrix φ_x is $n \times q$.

2.2. *Approximate Solutions.* Since the system (5) is generally nonlinear, approximate methods must be employed. In this connection, we introduce here the scalar performance indexes

$$P(x) = \varphi^T(x)\,\varphi(x), \qquad Q(x, \lambda) = F_x^T(x, \lambda)\,F_x(x, \lambda), \tag{6}$$

which measure the errors in the constraint and the optimum condition, respectively. Then, we observe that $P = 0$ and $Q = 0$ for the optimum solution, while $P > 0$ and/or $Q > 0$ for any approximation to the solution. When approximate methods are used, they must ultimately lead to values of x and λ such that

$$P(x) \leq P_\epsilon, \quad Q(x, \lambda) \leq Q_\epsilon. \tag{7}$$

Alternatively, (7) can be replaced by

$$R(x, \lambda) \leq R_\epsilon, \tag{8}$$

where

$$R(x, \lambda) = P(x) + Q(x, \lambda) \tag{9}$$

denotes the cumulative error in the constraint and the optimum result. In Eqs. (7)-(8), P_ϵ, Q_ϵ, R_ϵ are small, preselected numbers. Note that, if one chooses $P_\epsilon = Q_\epsilon = R_\epsilon$, satisfaction of Ineq. (8) implies satisfaction of Ineqs. (7).

3. Penalty Function Method

The penalty function method is based on the construction of a sequence of special functions having, in the limit, an unconstrained minimum point coincident with the solution of the original constrained minimization problem. Specifically, the penalty function is defined by

$$U(x, k) = f(x) + kP(x) = f(x) + k\varphi^T(x)\varphi(x) \tag{10}$$

and is obtained by adding to the function $f(x)$ a term quadratic in the constraint $\varphi(x)$, where $k > 0$ is the penalty constant.

The problem of minimizing the function (1) subject to the constraints (2) is replaced by a sequence of unconstrained minimization problems. In each element of the sequence or cycle, one minimizes the penalty function (10) with respect to x for given k. Therefore, theoretically speaking, the following necessary condition must be satisfied at the end of a cycle:

$$U_x(x, k) = f_x(x) + kP_x(x) = f_x(x) + 2k\varphi_x(x)\varphi(x) = 0, \tag{11}$$

If one defines the Lagrange multiplier to be

$$\lambda = 2k\varphi(x), \tag{12}$$

Eq. (11) can be rewritten as

$$F_x(x, \lambda) = f_x(x) + \varphi_x(x)\lambda = 0, \tag{13}$$

meaning that the combination of x and λ obtained at the end of a cycle satisfies exactly the optimum condition (5-2). However, if the penalty constant k is arbitrary, the vector x which satisfies Eq. (11) generally violates the constraint condition (6-1).

In order to obtain constraint satisfaction, increasingly larger values of the penalty constant must be employed in successive cycles of the penalty function method. In this connection, let k_1 denote the penalty constant of the present cycle and k_2 denote the penalty constant of the next cycle, with $k_2 > k_1$. Because of the jump in k, the penalty function increases by the amount

$$U(x, k_2) - U(x, k_1) = (k_2 - k_1)P(x), \tag{14}$$

and the norm squared of the gradient of the penalty function takes on the value[1]

$$U_x^T(x, k_2)\,U_x(x, k_2) = (k_2 - k_1)^2 P_x^T(x)\,P_x(x) = (\pi - 1)^2 f_x^T(x) f_x(x), \tag{15}$$

where

$$P(x) = \varphi^T(x)\varphi(x), \qquad P_x(x) = 2\varphi_x(x)\varphi(x). \tag{16}$$

The positiveness of the right hand side of Eq. (14) is the key to the mechanism on which the penalty function method is based.

After a sufficient number of cycles, the constraint error can be made as small as desired provided the penalty constant has become sufficiently large. Theoretically speaking, the condition $\varphi(x) = 0$ is desired at convergence; consequently, the multiplier λ defined by Eq. (12) can be identical with the multiplier satisfying Eqs. (5), which is generally nonzero, only if $k \to \infty$. In a practical digital computer, this means that large values of k are needed at convergence.

3.1. *Numerical Implementation.* From the above considerations, the following outline of the penalty function method emerges.

(a) The original constrained minimization problem is replaced by a sequence of unconstrained minimization problems.

(b) In each element of the sequence or cycle, the penalty function

$$U(x, k) = f(x) + k\varphi^T(x)\varphi(x) \tag{17}$$

is minimized with respect to x for given k. The minimum of $U(x, k)$ is achieved when the following stopping condition is satisfied:

$$U_x^T(x, k)\,U_x(x, k) \leq Q_x, \tag{18}$$

where Q_x is a small, preselected number.

(c) Upon termination of a cycle, one checks the following inequality:

$$\varphi^T(x)\varphi(x) + U_x^T(x, k)\,U_x(x, k) \leq R_x, \tag{19}$$

where R_x is a small, preselected number. If Ineq. (19) is satisfied, the algorithm is terminated. If Ineq. (19) violated, the algorithm is continued by choosing the solution point of the present cycle as the starting point of the next cycle.

---

[1] Note that $U_x^T(x, k_1)\,U_x(x, k_1) = 0$.

(d) For the next cycle, a higher value of the penalty constant is selected, specifically,

$$k_2 = \pi k_1, \tag{20}$$

where $\pi > 1$ is the penalty constant ratio. After updating the penalty constant, one returns to (b) and continues iteratively.

Remark 3.1. At convergence of a cycle, the stopping condition (18) can be written as

$$Q(x, \lambda) \leq Q_\epsilon, \tag{21}$$

where λ is given by Eq. (12). Analogously, at convergence of the algorithm, the stopping condition (19) becomes

$$R(x, \lambda) = P(x) + Q(x, \lambda) \leq R_\epsilon, \tag{22}$$

where λ is given by Eq. (12).

4. Updating Rules

Crucial to the penalty function method is the prediction of the ratio

$$\pi = k_2/k_1 \tag{23}$$

at which the penalty constant must be increased when shifting from one cycle of the algorithm to the next. Here, k_1 denotes the penalty constant of the present cycle, and k_2 denotes the penalty constant of the next cycle.

4.1. *Standard Technique.* The standard method (see, for instance, Ref. 7) consists of employing a constant value of π throughout the algorithm, for example,

$$\pi = 5 \quad \text{or} \quad \pi = 10. \tag{24}$$

4.2. *Proposed Technique.* An alternate method consists of employing a variable value of π throughout the algorithm. For instance, one might select π so as to achieve a predetermined constraint error at the end of the next cycle of the algorithm.

Let the following definition be introduced:

$$Z = \lambda^T \lambda, \tag{25}$$

with λ given by Eq. (12). Let Eqs. (12) and (25) be applied twice, once at the end of the present cycle (subscript 1) and once at the end of the next cycle (subscript 2). One obtains the relations

$$Z_1 = k_1^2 P_1, \qquad Z_2 = k_2^2 P_2, \tag{26}$$

which imply that

$$\pi = \sqrt{(Z_2 P_1 / Z_1 P_2)}. \tag{27}$$

Now, we introduce the basic assumption (*)

$$Z_1 = Z_2,$$

that is, we neglect the change in the norm squared of the multiplier between the end of the present cycle and the end of the next cycle. With this understanding, Eq. (27) simplifies to

$$\pi = \sqrt{(P_1/P_2)}. \tag{28}$$

This relation enables one to predict the penalty constant needed for the next cycle in order to achieve a predetermined constraint error P_2.

As an example, assume that the expected constraint error at the end of the next cycle P_2 is below the geometric mean of the constraint error at the end of the present cycle P_1 and that allowed for convergence P_ϵ; that is, assume that

$$P_2 = \sqrt{(P_1 P_\epsilon)}/\beta^2,$$

where $\beta > 1$. Then, Eq. (28) simplifies to

$$\pi = \beta \sqrt{(P_1/P_\epsilon)}. \tag{31}$$

As another example, assume that the expected constraint error at the end of the next cycle P_2 is below that allowed for convergence P_ϵ; that is, assume that

$$P_2 = P_\epsilon/\beta^2, \tag{32}$$

where $\beta > 1$. Then, Eq. (29) simplifies to

$$\pi = \beta \sqrt{(P_1/P_\epsilon)}. \tag{33}$$

(*) Hypothesis (28) is key to this paper and is supported by the data exhibited in Tables 1-4.

5. Experimental Conditions

In order to evaluate the theory, six numerical examples were solved using a Burroughs B-5500 computer, double-precision arithmetic, and FORTRAN-IV program. Within each cycle of the penalty function method, the conjugate gradient algorithm was employed (Ref. 1).

Starting Point. In all the examples, the nominal point chosen to start the penalty function method was defined by

$$x_1 = x_2 = \ldots = x_n = 2, \tag{34}$$

where $x_1, x_2, \ldots, x_n$ denote the components of the vector x.

Starting Penalty Constant. For the first cycle of the penalty function method, five values of the penalty constant were employed, namely,

$$k_0 = 10^{-1}, \quad k_0 = 10^{-4}, \quad k_0 = 10^{1}, \quad k_0 = 10^{4}, \quad k_0 = 10^{7}. \tag{35}$$

Updating Rules. For subsequent cycles, the penalty constant was determined in accordance with one of the following penalty constant ratios:

$$n = 5, \quad n = 10, \tag{36}$$

and

$$n = \beta \sqrt[4]{(P_1/P_0)}, \quad n = \beta \sqrt{(P_1/P_0)}, \tag{37}$$

with

$$\beta = 10, \quad P_0 = 10^{-10}. \tag{38}$$

The penalty constant ratios (36) are representative of the standard method (see, for instance, Ref. 7) and the penalty constant ratios (37) are representative of the new method proposed here.

Definition of Convergence. Convergence of a cycle was defined as

$$Q(x, k) = U_x^T(x, k)\, U_x(x, k) \leq 10^{-11}, \tag{39}$$

and convergence of the complete algorithm was defined as

$$R(x, k) = P(x) + Q(x, k) = \varphi^T(x)\, \varphi(x) + U_x^T(x, k)\, U_x(x, k) \leq 10^{-10}, \tag{40}$$

where the multiplier λ is given by Eq. (12).

Search Technique. Within each iteration of the conjugate gradient algorithm, let x denote the nominal point, $\bar{x}$ the varied point, and p the search direction. Let $\bar{U}(\alpha)$ denote the function representing the behavior of the penalty function along the search direction, that is,

$$\bar{U} = \bar{U}(\bar{x}, k) = U(x + \Delta x, k) = U(x - \alpha p, k) = \bar{U}(\alpha). \tag{41}$$

This function was employed in order to determine the optimum stepsize at each iteration. Specifically, a precise one-dimensional search on the function $\bar{U}(\alpha)$ was conducted such that, in any given step, the inequality (*)

$$\bar{U}(\alpha) < \bar{U}(\alpha_0) \tag{42}$$

was satisfied, where α_0 is the nominal stepsize and α is the varied stepsize. The search was terminated when the stopping condition

$$\bar{U}_\alpha'(\alpha) \leq \bar{U}_\alpha'(0) \times 10^{-4} \tag{43}$$

was satisfied.

6. Numerical Examples

In this section, six numerical examples are described. The first example pertains to a quadratic function subject to linear constraints. The remaining examples pertain to nonquadratic functions subject to nonlinear constraints.

EXAMPLE 6.1. Consider the problem of minimizing the function

$$f = (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2, \tag{44}$$

subject to the constraints

$$x_1 + 3x_2 = 0, \quad x_3 + x_4 - 2x_5 = 0, \quad x_2 - x_5 = 0. \tag{45}$$

(*) Due to the analytical nature of the examples considered here, multiple-point quasilinearization was employed in order to assure satisfaction of Ineq. (42); however, in a more realistic situation, one would use quadratic interpolation or cubic interpolation. For the details of the conjugate gradient algorithm and the search technique employed, the reader should consult Ref. 1.

This function admits the relative minimum $f = 4.0830$ at the point defined by

$$x_1 = -0.7674, \quad x_2 = 0.2558, \quad x_3 = 0.6279, \quad x_4 = -0.1162, \quad x_5 = 0.2558 \tag{46}$$

and

$$\lambda_1 = 2.0468, \quad \lambda_2 = 2.2328, \quad \lambda_3 = -5.0534. \tag{47}$$

Example 6.2. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^4 \tag{48}$$

subject to the constraint

$$x_1(1 + x_2^2) + x_3^4 - 4 - 3\sqrt{2} = 0. \tag{49}$$

This function admits the relative minimum $f = 0.3256 \times 10^{-1}$ at the point defined by

$$x_1 = 1.1048, \quad x_2 = 1.1906, \quad x_3 = 1.5352 \tag{50}$$

and

$$\lambda_1 = -0.1074 \times 10^{-1}. \tag{51}$$

Example 6.3. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^4 + (x_3 - x_4)^4 + (x_4 - x_5)^6 \tag{52}$$

subject to the constraints

$$x_1 + x_2^2 + x_3^4 - 2 - 3\sqrt{2} = 0, \quad x_2 - x_3^2 + x_4 + 2 - 2\sqrt{2} = 0,$$

$$x_1 x_5 - 2 = 0. \tag{53}$$

The function admits the relative minimum $f = 0.7877 \times 10^{-1}$ at the point defined by

$$x_1 = 1.1211, \quad x_2 = 1.3026, \quad x_3 = 1.4728, \quad x_4 = 1.6180, \quad x_5 = 1.6790 \tag{54}$$

and

$$\lambda_1 = -0.3882 \times 10^{-1}, \quad \lambda_2 = -0.1072 \times 10^{-1}, \quad \lambda_3 = -0.2879 \times 10^{-1}. \tag{55}$$

Example 6.4. Consider the problem of minimizing the function

$$f = 0.01(x_1 - 1)^2 + (x_2 - x_1^2)^2 \tag{56}$$

subject to the inequality constraint

$$x_1 \leq -1. \tag{57}$$

Introduce the auxiliary variable x_3 defined by

$$x_1 + x_3^2 + 1 = 0. \tag{58}$$

Then, the previous problem can be recast as that of minimizing the function (56) subject to the equality constraint (58). The function (56) admits the relative minimum $f = 0.04$ at the point defined by

$$x_1 = -1, \quad x_2 = 1, \quad x_3 = 0 \tag{59}$$

and

$$\lambda_1 = 0.04. \tag{60}$$

Example 6.5. Consider the problem of minimizing the function

$$f = -x_1 \tag{1}$$

subject to the inequality constraints

$$x_2 \geq x_3^2, \quad x_2 \leq x_3^3.$$

Introduce the auxiliary variables x_4 and x_5 defined by

$$x_2 - x_3^2 - x_4^2 = 0, \quad x_2 - x_3^3 - x_5^2 = 0. \tag{63}$$

Then, the previous problem can be recast as that of minimizing the function (61) subject to the equality constraints (63). The function (61) admits a relative minimum $f = -1$ at the point defined by

$$x_1 = 1, \quad x_2 = 1, \quad x_3 = 0, \quad x_4 = 0 \tag{64}$$

and

$$\lambda_1 = -1, \quad \lambda_2 = -1. \tag{65}$$

Example 6.6. Consider the problem of minimizing the function

$$f = \log x_1 - x_2$$

subject to the equality constraint

$$x_1^2 + x_2^2 - 4 = 0 \tag{61}$$

and the inequality constraint

$$x_3 \geq 1. \tag{68}$$

Introduce the auxiliary variable x_4 defined by

$$x_3 = 1 + x_4^2. \tag{69}$$

Then, the previous problem can be recast as that of minimizing the function

$$f = \log (1 + x_2^2) - x_1 \tag{70}$$

subject to the equality constraint

$$(1 + x_2^2)^2 + x_1^2 - 4 = 0. \tag{71}$$

Note that x_4 has been eliminated from the problem and can be computed *a posteriori* with (69). The function (70) admits the relative minimum $f = -\sqrt{3}$ at the point defined by

$$x_1 = 0, \quad x_2 = \sqrt{3}, \quad x_3 = 1 \tag{72}$$

and

$$\lambda_1 = 1/2\sqrt{3}. \tag{73}$$

7. Results and Conclusions

The examples described in Section 6 were solved with the penalty function method in conjunction with the conjugate gradient algorithm according to the experimental conditions outlined in Section 6. The numerical results are presented in Tables 1-16. Tables 1-4 supply a justification of the basic assumption (28), and Tables 5-16 supply comparative data for the standard updating rules and the new updating rules.

Basic Hypothesis. Tables 1-4 refer to a particular example, namely, Example 6.5. They show the behavior of the penalty function method for each of the penalty constant ratios (36) and (37) assuming that the starting penalty constant is $k_0 = 1$. In the tables, the quantities P, Q, R, Z are shown versus the iteration number N and the cycle number N_c. To shorten the tables, these quantities are given only at the initial point and the final point of each cycle.

As the tables indicate, the stopping condition (32) is met at convergence of a cycle and the stopping condition (40) is met at convergence of the algorithm. Of particular interest is the behavior of the quantity Z, defined by $Z = P^2/I = 4P^2/P$. If the end conditions of successive cycles are compared, it is clear that Z approaches an asymptotic value as the cycle number N_c increases and the penalty constant k becomes sufficiently large. This asymptotic value is independent of the particular penalty constant ratio employed and is the same in each of Tables 1-4, namely $Z = 3$. This is in agreement with the theoretical values of the multipliers at convergence, which are supplied by Eqs. (68). This result confirms the basic assumption (28) and justifies the reasoning leading to the general penalty constant ratio (20).

Comparative Data. Tables 5-16 refer to Examples 6.1 through 6.6. They show the number of iterations at convergence N, and the number of cycles at convergence N_c, versus the starting penalty constant k_0 for each of the penalty constant ratios (30)-(37). Since each iteration requires the same amount of computational work, it is clear that the number of iterations is representative of the computer time required for convergence.

For all of the examples and each starting penalty constant k_0, the variable-ratio updating rules are superior to the constant-ratio updating rules in that they require a smaller number of iterations for convergence. This smaller number of iterations is achieved by employing a smaller number of cycles and by increasing the penalty constant at a higher rate than that given by Eqs. (36).

TABLE 1. Example 6.5, $k_0 = 1$, $n = 5$.

N_0	N	h	P	Q	R	S
1	9	0.10 E + 01	0.10 E + 03	0.57 E + 05	0.57 E + 05	0.43 E + 03
1	17	0.10 E + 01	0.05 E − 01	0.12 E − 15	0.93 E − 01	0.38 E + 00
2	17	0.50 E + 01	0.05 E − 01	0.18 E + 03	0.18 E + 03	0.95 E + 01
2	27	0.50 E + 01	0.02 E − 03	0.12 E − 13	0.02 E − 03	0.02 E + 00
3	27	0.25 E + 02	0.02 E − 03	0.16 E + 03	0.16 E + 03	0.23 E + 03
3	31	0.25 E + 02	0.62 E − 06	0.14 E − 14	0.62 E − 03	0.16 E + 01
4	37	0.13 E + 03	0.62 E − 06	0.18 E + 03	0.16 E + 03	0.39 E + 03
4	47	0.13 E + 03	0.50 E − 04	0.60 E − 18	0.30 E − 04	0.19 E + 01
5	47	0.63 E + 03	0.50 E − 04	0.16 E + 03	0.16 E + 03	0.47 E + 03
5	63	0.63 E + 03	0.13 E − 05	0.13 E − 12	0.13 E − 05	0.20 E + 01
6	63	0.31 E + 04	0.13 E − 05	0.16 E + 03	0.16 E + 03	0.45 E + 03
6	69	0.31 E + 04	0.61 E − 07	0.84 E − 17	0.61 E − 07	0.20 E + 01
7	69	0.16 E + 05	0.31 E − 07	0.16 E + 03	0.16 E + 03	0.50 E + 03
7	85	0.16 E + 05	0.30 E − 08	0.14 E − 16	0.30 E − 08	0.20 E + 01
8	85	0.78 E + 05	0.30 E − 08	0.16 E + 03	0.16 E + 03	0.50 E + 03
8	76	0.78 E + 05	0.15 E − 10	0.35 E − 13	0.15 E − 10	0.20 E + 01
9	76	0.39 E + 06	0.92 E − 10	0.16 E + 03	0.16 E + 03	0.50 E + 03
9	75	0.39 E + 06	0.35 E − 11	0.84 E − 17	0.35 E − 11	0.20 E + 01
10	75	0.20 E + 07	0.38 E − 11	0.16 E + 03	0.16 E + 03	0.50 E + 03
10	10	0.20 E + 07	0.18 E − 12	0.17 E − 13	0.18 E − 12	0.20 E + 01

TABLE 2. Example 6.5, $k_0 = 1$, $n = 10$.

N_0	N	h	P	Q	R	S
1	9	0.10 E + 01	0.10 E + 03	0.57 E + 05	0.61 E + 05	0.43 E + 03
1	17	0.10 E + 01	0.05 E − 01	0.12 E − 15	0.05 E − 01	0.38 E + 00
2	17	0.10 E + 02	0.05 E − 01	0.61 E + 03	0.61 E + 03	0.95 E + 02
2	30	0.20 E + 02	0.30 E − 03	0.14 E − 13	0.30 E − 03	0.19 E + 01
3	30	0.10 E + 03	0.30 E − 03	0.61 E + 03	0.61 E + 03	0.12 E + 03
3	40	0.10 E + 03	0.45 E − 06	0.73 E − 14	0.45 E − 01	0.19 E + 01
4	40	0.50 E + 03	0.45 E − 06	0.61 E + 03	0.61 E + 03	0.19 E + 03
4	47	0.50 E + 03	0.50 E − 08	0.50 E − 14	0.60 E − 08	0.20 E + 01
5	47	0.10 E + 04	0.50 E − 08	0.61 E + 03	0.61 E + 03	0.20 E +
5	69	0.10 E + 04	0.60 E − 08	0.15 E − 17	0.60 E − 08	0.20 E +
6	69	0.10 E + 05	0.30 E − 08	0.61 E + 03	0.61 E + 03	0.20 E + 03
6	69	0.10 E + 05	0.60 E − 10	0.21 E − 20	0.60 E − 10	0.20 E + 01
7	80	0.10 E + 01	0.60 E − 10	0.61 E + 03	0.61 E + 03	0.20 A + 03
7	84	0.50 E + 07	0.60 E − 12	0.19 E − 18	0.60 E − 12	0.20 E + 01

Comparison of Multiplier and Quasilinearization Methods

Alejandro V. Levy*

CINAS, Universidad Nacional Autónoma de México, Mexico City, Mexico and Department of Mechanical Engineering, Rice University, Houston, Texas

Antonio Montalvo

Department of Mathematics, Universidad Iberoamericana, Mexico City, Mexico

Miele et al. (1972) developed the method of multipliers for minimizing a function $f(x)$ subject to the constraint $\varphi(x) = 0$, where f is a scalar, x is an n vector, and φ is a q vector, with $q < n$. In this paper, a comparison of the standard quasilinearization method and the method of multipliers is presented. The comparison is made by considering the relative stability and the relative speed of convergence of the two methods. Numerical results are presented for nine examples, each of which is solved for 100 different starting points. The examples show that, in general, the method of multipliers is a more economic and robust algorithm than the standard quasilinearization method. In particular, this advantage increases in proportion to the following characteristics of the problem being solved: (a) the ratio q/n and the size of the problem and (b) the nonlinearity of the problem and the number of relative minima and maxima that may exist.

1. Introduction

In recent years, considerable attention has been given to the quasilinearization methods for minimizing a function $f(x)$ subject to a constraint $\varphi(x) = 0$. Here, f is a scalar, x is an n-vector, and φ is a q-vector, with $q < n$.

In the *standard quasilinearization algorithm* (SQL), the augmented function $F(x,\lambda) = f(x) + \lambda^T\varphi(x)$ is utilized, where λ, a q-vector, is an unknown Lagrange multiplier. Starting from nominal values of x and λ, the method obtains corrections Δx and $\Delta\lambda$ by solving a linear system of equations. In this way, updated values $\bar{x} = x + \Delta x$ and $\bar{\lambda} = \lambda + \Delta\lambda$ are obtained. Then, the process is repeated iteratively until convergence occurs. For the particular case of a quadratic function subject to a linear constraint, the method converges to the solution in one iteration. However, for a nonlinear function subject to nonlinear constraints, the method is rather unstable. Depending on the nominal values of x and λ, it can lead to a relative minimum, a relative maximum, or an inflection point; occasionally, the method produces displacements which are so purposeless that overflow can occur in a given computer.

In the *modified quasilinearization algorithm* (MQL), Miele et al. (1973), the same basic approach is taken as in SQL, with one important modification: the stepsize α, $0 < \alpha \leq 1$, is introduced in order to control the magnitude of the displacements. The result is an improved quasilinearization algorithm that (i) retains quadratic convergence for a linear-quadratic problem and (ii) improves the stability of SQL for nonlinear functions subject to nonlinear constraints.

Both the above mentioned methods require the solution of a linear system of equations of order $n + q$. Thus, if Gaussian elimination is employed, the computational effort is of the order of $(n + q)^3$ multiplications.

For large systems, one way to reduce the computational effort per iteration is to resort to penalty function methods, either in the standard format or the multiplier format. Here, the unconstrained minimization of the augmented penalty function $W(x,\lambda,k) = F(x,\lambda) + k\varphi^T(x)\varphi(x)$, where the scalar k is the penalty constant.

With reference to the method of multipliers, an important modification was presented by Miele et al. (1972). In this modification, called the *modified method of multipliers* (MMM), the augmented penalty function is employed in connection with several minimization algorithms, namely, the ordinary gradient algorithm, the conjugate gradient algorithm, and the modified quasilinearization algorithm. Improved updating rules for the Lagrange multiplier and the penalty constant were developed in connection with each minimization algorithm.

When the modified method of multipliers is employed in conjunction with the modified quasilinearization algorithm, a linear system of equations of order n must be solved at every iteration. Therefore, if Gaussian elimination is employed, the computational effort is of the order of n^3 multiplications.

Besides reducing the computational effort per iteration, MMM is more stable than SQL. This is because MMM has a descent property on the augmented penalty function $W(x,\lambda,k)$. While SQL might converge to a relative minimum, a relative maximum, or an inflection point, MMM generally leads to a relative minimum.

The standard quasilinearization algorithm was used by Luus and Jaakola (1973) with some success to solve several problems, using random numbers as starting values for x and λ. The purpose of this report is to show that even greater success and computer time savings can be obtained if the modified method of multipliers is utilized instead of the standard quasilinearization algorithm.

In section 2, the statement of the problem is given together with the first-order optimality conditions. In sections 3 and 4, the standard quasilinearization algorithm and the modified method of multipliers are reviewed briefly. In section 5, a comparison of the computational effort per iteration is given. In sections 6 and 7, the experimental conditions and nine numerical examples are presented. In section 8, numerical results are discussed, and the conclusions are stated.

2. Statement of the Problem

We consider the problem of minimizing the function

$$f = f(x) \tag{1}$$

subject to the constraint

$$\varphi(x) = 0 \tag{2}$$

TABLE 3. Example 6.5, $k_0 = 1$, $\pi = 10\sqrt[4]{(P_1/P_0)}$.

N_0	N	k	P	Q	R	S
1	0	0.10 E + 01	0.10 E + 03	0.57 E + 05	0.57 E + 05	0.43 E + 03
1	17	0.10 E + 01	0.95 E − 01	0.13 E − 15	0.95 E − 01	0.98 E + 00
2	17	0.55 E + 04	0.95 E − 01	0.31 E + 08	0.31 E + 08	0.13 E + 03
2	32	0.55 E + 04	0.16 E − 07	0.89 E − 12	0.16 E − 07	0.90 E + 01
3	32	0.63 E + 06	0.16 E − 07	0.13 E + 06	0.13 E + 05	0.25 E + 03
3	48	0.63 E + 06	0.13 E − 11	0.35 E − 15	0.13 E − 11	0.10 E + 01
4	48	0.67 E + 07	0.13 E − 11	0.93 E + 03	0.93 E + 02	0.29 E + 03
4	43	0.67 E + 07	0.11 E − 15	0.34 E − 19	0.35 E − 19	0.20 E + 01

TABLE 4. Example 6.5, $k_0 = 1$, $\pi = 10\sqrt[4]{(P_1/P_0)}$.

N_0	N	k	P	Q	R	S
1	0	0.10 E + 01	0.10 E + 03	0.57 E + 05	0.57 E + 05	0.43 E + 03
1	17	0.10 E + 01	0.95 E − 01	0.13 E − 15	0.95 E − 01	0.98 E + 00
2	17	0.31 E + 07	0.95 E − 01	0.95 E + 13	0.95 E + 13	0.34 E + 13
2	34	0.31 E + 07	0.58 E − 19	0.60 E − 16	0.63 E − 19	0.20 E + 01

TABLE 5. Example 6.1, number of iterations N_0.

k_0	$\pi = 5$	$\pi = 10$	$\pi = 10\sqrt[4]{(P_1/P_0)}$	$\pi = 10\sqrt[4]{(P_0/P_1)}$
10^{-3}	70	69	34	33
10^{-2}	60	47	31	17
10^{-1}	58	43	31	19
10^{0}	48	37	33	29
10^{1}	46	33	36	25

TABLE 6. Example 6.1, number of cycles N_0.

k_0	$\pi = 5$	$\pi = 10$	$\pi = 10\sqrt[4]{(P_1/P_0)}$	$\pi = 10\sqrt[4]{(P_0/P_1)}$
10^{-3}	14	10	8	8
10^{-2}	12	9	4	3
10^{0}	11	6	4	3
10^{1}	10	7	4	3
10^{2}	8	6	4	3

TABLE 7. Example 6.2, number of iterations N_ϵ.

ϵ_0	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt{(F_1/F_0)}$	$\alpha = 10\sqrt{(F_2/F_0)}$
10^{-2}	62	62	50	41
10^{-1}	54	60	50	45
10^{0}	58	60	56	65
10^{1}	64	67	54	61
10^{2}	80	80	77	76

TABLE 8. Example 6.2, number of cycles N_{ϵ_c}.

ϵ_0	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt{(F_1/F_0)}$	$\alpha = 10\sqrt{(F_2/F_0)}$
10^{-2}	10	7	4	2
10^{-1}	8	6	3	2
10^{0}	7	5	3	2
10^{1}	6	4	4	2
10^{2}	4	5	5	2

TABLE 9. Example 6.3, number of iterations N_ϵ.

ϵ_0	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt{(F_1/F_0)}$	$\alpha = 10\sqrt{(F_2/F_0)}$
10^{-2}	(a)	(a)	134	174
10^{-1}	154	163	120	88
10^{0}	156	163	99	106
10^{1}	135	139	83	76
10^{2}	138	131	107	103

TABLE 10. Example 6.3, number of cycles N_{ϵ_c}.

ϵ_0	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt{(F_1/F_0)}$	$\alpha = 10\sqrt{(F_2/F_0)}$
10^{-2}	(a)	(a)	4	3
10^{-1}	8	7	4	3
10^{0}	8	6	3	3
10^{1}	6	5	3	3
10^{2}	5	4	3	3

(a) Number of iterations exceeded 300.

TABLE 11. Example 6.4, number of iterations N_{it}.

λ_0	$n=5$	$n=10$	$n=10\sqrt[4]{(P_1/P_0)}$	$n=10\sqrt[4]{(P_2/P_0)}$
10^{-2}	95	68	35	36
10^{-1}	84	47	25	31
10^{0}	58	45	29	35
10^{1}	81	78	70	68
10^{2}	138	170	187	164

TABLE 12. Example 6.4, number of cycles N_{cy}.

λ_0	$n=5$	$n=10$	$n=10\sqrt[4]{(P_1/P_0)}$	$n=10\sqrt[4]{(P_2/P_0)}$
10^{-2}	13	8	4	3
10^{-1}	9	7	3	3
10^{0}	5	4	3	3
10^{1}	8	6	3	3
10^{2}	8	4	3	3

TABLE 13. Example 6.5, number of iterations N_{it}.

λ_0	$n=5$	$n=10$	$n=10\sqrt[4]{(P_1/P_0)}$	$n=10\sqrt[4]{(P_2/P_0)}$
10^{-2}	115	89	61	44
10^{-1}	98	78	47	37
10^{0}	79	64	49	54
10^{1}	87	61	58	50
10^{2}	53	48	35	31

TABLE 14. Example 6.5, number of cycles N_{cy}.

λ_0	$n=5$	$n=10$	$n=10\sqrt[4]{(P_1/P_0)}$	$n=10\sqrt[4]{(P_2/P_0)}$
10^{-2}	13	9	4	3
10^{-1}	11	8	4	3
10^{0}	10	7	4	3
10^{1}	8	8	3	3
10^{2}	7	8	3	3

TABLE 15. Example 6.6, number of iterations N_c.

ϵ_0	$n=6$	$n=10$	$n=10\sqrt[4]{(P_1/P_0)}$	$n=10\sqrt[4]{(P_2/P_0)}$
10^{-2}	27	21	13	10
10^{-1}	23	18	12	8
10^{0}	21	19	12	11
10^{1}	20	16	13	11
10^{2}	26	24	20	18

TABLE 16. Example 6.6, number of cycles N_{cy}.

ϵ_0	$n=6$	$n=10$	$n=10\sqrt[4]{(P_1/P_0)}$	$n=10\sqrt[4]{(P_2/P_0)}$
10^{-2}	12	9	4	3
10^{-1}	10	8	4	3
10^{0}	8	7	3	3
10^{1}	7	6	3	3
10^{2}	6	6	3	3

REFERENCES

[1] MIELE, A., COGGINS, G. M., and LEVY, A. V., "*Updating Rules for the Penalty Constant Used in the Penalty Function Method for Mathematical Programming Problems*," Rice University, Aero-Astronautics Report No. 60, 1972.

[2] MIELE, A., HUANG, H. Y., and HEIDEMAN, J. C., "*Sequential Gradient-Restoration Algorithm for the Minimization of Constrained Functions, Ordinary and Conjugate Gradient Versions*," Journal of Optimization Theory and Applications, Vol. 4, No. 4, 1969.

[3] MIELE, A., LEVY, A. V., and CRAGG, E. E., "*Modifications and Extensions of the Conjugate Gradient-Restoration Algorithm for Mathematical Programming Problems*," Journal of Optimization Theory and Applications, Vol. 7, No. 6, 1971.

[4] MIELE, A., HEIDEMAN, J. C., and LEVY, A. V., "*Combined Conjugate Gradient-Restoration Algorithm for Mathematical Programming Problems*," Ricerche di Automatica, Vol. 2, No. 2, 1971.

[5] KELLEY, H. J., "*Methods of Gradients*," Optimization Techniques, Edited by G. Leitmann, Academic Press, New York, 1962.

[6] BRYSON, A. E., Jr., and HO, Y. C., "*Applied Optimal Control*," Blaisdell Publishing Company, Waltham, Massachusetts, 1969.

[7] FIACCO, A. V., and MCCORMICK, G. P., "*Nonlinear Programming: Sequential Unconstrained Minimization Techniques*," John Wiley and Sons, New York, 1968.

[8] DAVIES, D., and SWANN, W. H., "*Review of Constrained Optimization*," Optimization, Edited by R. Fletcher, Academic Press, New York, 1969.

Paper No.
76-DE-E

an ASME publication

$3.00 PER COPY $1.50 TO ASME MEMBERS

G. A. Gabriele
Graduate Assistant.

K. M. Ragsdell
Associate Professor.
Mem. ASME

School of Mechanical Engineering,
Purdue University,
West Lafayette, Ind.

The Generalized Reduced Gradient Method: A Reliable Tool for Optimal Design

This paper is a presentation of a method, called the Generalized Reduced Gradient Method, which has not received wide attention in the engineering design literature. Included is a theoretical development of the method, a description of the basic algorithm, and additional recommendations to produce an efficient code. A Fortran code employing this theory was written and tested on the Eason and Fenton [1][1] test problems, illustrating the method to be efficient and reliable.

Introduction

Engineering design is a multiphase process requiring constant decision making by the designer. Basing decisions on his knowledge and experience, he must arrive at a combination of design variables that best satisfies his objectives. As engineering design has matured so have the guidelines and methods that the designer has at his disposal to aid him in his choices. Although his experience may provide some answers, many of his choices are based on analytical methods. Designers have generally been receptive to new methods, when they simplify and/or otherwise enhance the overall synthesis process. The availability of high-speed digital computing power has encouraged the application of modern optimization methods (mathematical programming methods) to engineering design. This marriage of efforts is generating developments in both fields, and has produced an important area of research known as optimal design.

Drawing on his experience the engineer is able to define variables, a design objective, and a set of constraints that must be met in order that the design be a workable solution. By developing corresponding equations the design problem can be formulated into a standard form acceptable to mathematical programming techniques. This standard form is defined here

Minimize

$$f(\mathbf{x}) \quad \mathbf{x} = [x_1, x_2, x_3, \ldots, x_N]^T, \quad \mathbf{x} \in R^N \tag{1}$$

subject to

$$\phi_k(\mathbf{x}) \geq 0 \quad k = 1, 2, 3, \ldots, K \tag{2}$$

$$\psi_l(\mathbf{x}) = 0 \quad l = 1, 2, 3, \ldots, L \tag{3}$$

where

$\mathbf{x}$ = a column vector of design variables
N = total number of design variables
$f(\mathbf{x})$ = design criterion or objective function
$\phi_k(\mathbf{x})$ = K inequality constraint functions; these functions delimit regions in the design space
$\psi_l(\mathbf{x})$ = L equality constraint functions; these functions vastly reduce the number of candidate designs, because they require specific combinations of the design variables

When equations (1-3) are linear functions in the design variables the problem falls in the general class of linear programming (LP) problems. Much of the research in the past 20 years in the field of mathematical programming has been with linear programming. The simplex algorithm has become highly developed and most linear programs can be solved using this technique.

A more general occurrence in engineering design arises when expressions (1-3) are nonlinear. This is known as the nonlinear programming (NLP) problem. No general method has been developed to solve nonlinear problems in the sense that the simplex algorithm exists to solve the linear problem. Although many strategies have been suggested, comparative studies [1, 2] have shown that no method has been successfully applied to all problems.

One procedure for solving the NLP that has seen widespread use is the penalty function approach. Consider the following function

$$P(\mathbf{x}) = f(\mathbf{x}) + W(r, R, \phi, \psi) \tag{4}$$

which we call the generalized weighted penalty function. $W(r, R, \phi, \psi)$ is the penalty term, and is constructed in such a manner that infeasible points (points satisfying (2) and (3)) are given heavy treatment. The basic concept of the penalty function method is that successive minimizations of the unconstrained function $P(\mathbf{x})$ will converge to the constrained minimum of the original NLP. A survey of penalty function techniques is given by Fiacco and McCormick [3].

Contributed by the Design Engineering Division for publication in the JOURNAL OF ENGINEERING FOR INDUSTRY. Manuscript received at ASME Headquarters June 1, 1976. Paper No. 76-DE-E.

[1] Numbers in brackets designate References at end of paper.

An advantage of penalty function methods is their ease of implementation. Any unconstrained searching technique can be used to perform the successive minimizations of $P(x)$. However, there are several difficulties with penalty function methods that do not make them a generally reliable technique. First, if the successive minimizations of $P(x)$ are not trivial, then the time to solution may become impractical. Penalty function methods are also very sensitive to parameter adjustment and constraint scaling, and may require repetitive adjustment to obtain convergence to an optimum. Finally, penalty function methods generally do not possess the ability to move along equality constraints. Rather they tend to get caught on a point that satisfies the constraints, but is not the constrained optimum.

Because the state of the art for linear programming is so well developed, another approach to solving the constrained NLP is linearization of the NLP to tailor it to the LP methods. Basically this can be accomplished by replacing the nonlinear functions of (1)–(3) by first-order Taylor series approximations and solving the resulting LP. The Griffith and Stewart method [4] is based on this approach.

The advantage of this approach lies in being able to use existing linear programming theory. The method performs well on large-scale problems when the objective function and constraints are nearly linear. Difficulties arise when these functions are very nonlinear. Hence in order that the linear approximations hold, only small changes in the design variables are allowed, which may cause progress to the optimum to be slow. Indeed in the presence of highly nonlinear functions, the LP solution may give rise to a poor search direction, therefore hindering the optimization process.

Other methods of handling the constrained NLP are described in the literature [5, 6]; however, many of these methods do not solve the complete NLP described in (1)–(3). The two approaches described here do handle the complete problem and have seen wide acceptance in optimal engineering design today. In this paper another approach known as the Reduced Gradient Method will be described. The Reduced Gradient Method avoids many of the problems associated with penalty function and LP-like methods, possessing one of the most powerful methods currently known for handling the constrained nonlinear programming problem.

Reduced Gradient—Theory

The Reduced Gradient Method was originally given by Wolfe for a nonlinear objective function with linear constraints [7, 8]. A generalization of Wolfe's method to accommodate nonlinearities in both the objective function and constraints was first accomplished by Abadie [9]. Concurrently to both Wolfe and Abadie, Wilde and Beightler developed their differential algorithm based on the constrained derivative [10]. The constrained derivative and the reduced gradient employ much the same theoretical basis, but for purposes of this discussion the method shall be known as the Reduced Gradient Method.

The general constrained nonlinear programming problem of (1)–(3) can be restated in the following form:

Minimize

$$f(x); \quad x = [x_1, x_2, \ldots, x_N]^T, \quad x \in R^N \tag{5}$$

subject to

$$\phi_m(x) \geq 0 \qquad m = 1, 2, \ldots, M \tag{6}$$

$$A \leq x \leq B \tag{7}$$

The $N \times 1$ vectors A and B represent upper and lower bounds on the design vector x. The inequality constraints have been included as equality constraints by using the following transformations:

$$\phi_k(x) = \phi_k(x) - S_k = 0$$

$$0 \leq S_k \leq \infty \qquad k = 1, 2, \ldots, K \tag{8}$$

The variables S_k are nonnegative slack variables which will be included in the original set of design variables. Hence the parameter N represents the total number of design variables plus the number of slack variables used in the transformation of (8). The parameter M represents the total number of constraints,

$$M = L + K \tag{9}$$

It should be stressed that the constraints of (6) are most trivial constraints; that is, functional constraints. Variable bounds are defined in (7) and will require separate handling.

Consider the following strategy. Divide the design vector of equation (9) into two classes which shall be known as the decision and state variables.

$$x = [z, y]^T \tag{10}$$

$$z = [z_1, z_2, z_3, \ldots, z_Q]^T; \quad \text{decision variables} \tag{11}$$

$$y = [y_1, y_2, y_3, \ldots, y_M]^T; \quad \text{state variables} \tag{12}$$

$$Q = N - M \tag{13}$$

The decision variables are completely independent, and the state variables are slaves to the decision variables used to satisfy the constraints $\phi_m(x)$.

We can see a motivation for this division of the x vector into decision and state variables based on the elimination technique for handling equality constraints. Given the equality constrained NLP,

Minimize

$$f(x); \quad x = [x_1, x_2, x_3, \ldots, x_N]^T \tag{14}$$

subject to

$$\psi_l(x) = 0 \qquad l = 1, 2, 3, \ldots, L \tag{16}$$

Reformulate each constraint in (16) to explicitly define a different design variable.

$$x_l = h_l(x) \qquad l = 1, 2, 3, \ldots, L \tag{16}$$

Substitution into the objective function (14) would yield

$$f(h_1, h_2 \ldots h_L, x_{L+1}, \ldots, x_N) \tag{17}$$

This new objective function becomes an unconstrained function in $N - L$ design variables. Minimization of this new objective function would yield the constrained optimal values of the $N - L$ design variables describing (17) with the remaining design variables calculated by backsubstituting into (16).

In the foregoing technique the decision variables became the $N - L$ design variables describing (17) with the state variables described by (16). We can see here a basic approach for constrained optimization

Nomenclature

$f(x), f$ = objective function
$x = x_1, x_2, \ldots, x_N$; set of design variables
R^N = N-dimensional space of reals
$\phi_k(x), \phi$ = inequality constraint functions
$\psi_l(x), \psi$ = equality constraint functions
N = number of design variables
K = number of inequality constraints
L = number of equality constraints
$P(x)$ = penalty function
$W(x, R, \phi, \psi)$ = penalty term
S_k = slack variables
M = total number of constraints
Q = number of decision variables
z = Q-dimensional column vector of decision variables
y = M-dimensional column vector of state variables
$L(x, u, w)$ = Lagrangian function
D, D_{ij} = transformation matrix, $M \times Q$
G = reduced gradient, $Q \times 1$
$P(z)$ = search direction of decision variables, $Q \times 1$
$P(y)$ = search direction providing first approximation of state variables, $M \times 1$
α = step length parameter
y^1 = first approximation of state variables
y^k = adjusted values of state variables
N_f = number of objective function evaluations
N_c = number of constraint set evaluations

ining the decision and state dichotomy. We can perturb the decision variables in the newly formed unconstrained space to seek the minimum of the objective function while adjusting the state variables to maintain feasibility. However, use of the technique just described presupposes that the state variables can be defined explicitly in terms of the constraint functions. In engineering design problems this is generally the exception and not the rule. Some other method must be devised to utilize the decision and state variable dichotomy when nonlinearity is a factor.

The following notation will be useful in the discussion to follow

$$g(x) = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_Q}\right]^T$$

$$g(y) = \left[\frac{\partial f(x)}{\partial y_1}, \frac{\partial f(x)}{\partial y_2}, \ldots, \frac{\partial f(x)}{\partial y_M}\right]^T$$

$$\frac{\partial \psi}{\partial x} = \begin{bmatrix} \frac{\partial \psi_1}{\partial x_1} & \frac{\partial \psi_1}{\partial x_2} & \cdots & \frac{\partial \psi_1}{\partial x_Q} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \frac{\partial \psi_M}{\partial x_1} & \frac{\partial \psi_M}{\partial x_2} & \cdots & \frac{\partial \psi_M}{\partial x_Q} \end{bmatrix} \quad (M \times Q)$$

$$\frac{\partial \psi}{\partial y} = \begin{bmatrix} \frac{\partial \psi_1}{\partial y_1} & \frac{\partial \psi_1}{\partial y_2} & \cdots & \frac{\partial \psi_1}{\partial y_M} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \frac{\partial \psi_M}{\partial y_1} & \frac{\partial \psi_M}{\partial y_2} & \cdots & \frac{\partial \psi_M}{\partial y_M} \end{bmatrix} \quad (M \times M)$$

Let us examine the first variation of $f(x)$ and $\psi(x)$,

$$df = g(x)^T dx + g(y)^T dy \tag{18}$$

$$d\psi = \frac{\partial \psi}{\partial x} dx + \frac{\partial \psi}{\partial y} dy = 0 \tag{19}$$

where

$dx = Q \times 1$ vector of differential displacements of x

$dy = M \times 1$ vector of differential displacements of y

Solving (19) for dy yields,

$$dy = -\frac{\partial \psi}{\partial y}^{-1} \frac{\partial \psi}{\partial x} dx \tag{20}$$

Substituting (20) into (18) and rearranging will yield the following linear approximation to the *reduced gradient*:

$$g_r(x)^T = g(x)^T - g(y)^T \frac{\partial \psi}{\partial y}^{-1} \frac{\partial \psi}{\partial x} \tag{21}$$

The reduced gradient defines the rate of change of the objective function with respect to the decision variables with the state variables adjusted to maintain feasibility. Expression (20) gives the changes necessary in the states for a given change in the decisions for linear constraints. Geometrically the reduced gradient can be described as a projection of the original N-dimensional gradient onto the $(N - M)$-dimensional feasible region described by the decision variables.

A necessary condition for the existence of a minimum of an unconstrained nonlinear function is that the elements of the gradient vanish. Similarly, a minimum of the constrained nonlinear function occurs when the appropriate elements of the reduced gradient vanish. This conclusion can be verified by a comparison with the Kuhn-Tucker [11] conditions for the existence of a constrained relative minimum.

By first transforming the variable bounds into inequality constraints,

$$\phi_i(x) = x_i - a_i \geq 0 \quad \left.\begin{matrix} \\ \end{matrix}\right| \quad i = 1, 2, 3, \ldots, N \tag{22}$$

$$\phi_{N+i}(x) = b_i - x_i \geq 0 \quad \left.\begin{matrix} \\ \end{matrix}\right| \tag{23}$$

we can form the following Lagrangian function,

$$L(x, u, v) = f(x) + \sum_{m=1}^{M} v_m \psi_m(x) - \sum_{j=1}^{2N} u_j \phi_j(x) \tag{24}$$

The following Kuhn-Tucker necessary conditions hold for a point to be a relative minimum x^*,

$$g(x^*)^T + \sum_{m=1}^{M} v_m^* \frac{\partial \psi_m}{\partial x} - \sum_{j=1}^{J} u_j^* \frac{\partial \phi_j}{\partial x} = 0 \tag{25}$$

and

$$\psi_m(x^*) = 0 \qquad m = 1, 2, \ldots, M \tag{26}$$

$$\phi_j(x^*) \geq 0 \qquad j = 1, 2, \ldots, J = 2N \tag{27}$$

$$u_j^* \phi_j(x^*) = 0 \qquad j = 1, 2, \ldots, J = 2N \tag{28}$$

$$u_j^* \geq 0 \qquad j = 1, 2, \ldots, J = 2N \tag{29}$$

$$v_m^* \gtrless 0 \qquad m = 1, 2, \ldots, M \tag{30}$$

Introducing decision and state variables into (25) and decomposing we can obtain the following form:

$$g(x^*)^T + v^{*T} \frac{\partial \psi}{\partial x^*} - u^{*T} \frac{\partial \phi}{\partial x^*} = 0 \tag{31}$$

$$g(y^*)^T + v^{*T} \frac{\partial \psi}{\partial y^*} - u^{*T} \frac{\partial \phi}{\partial y^*} = 0 \tag{32}$$

For reasons to be examined in the next section, a state variable is not allowed to be equal or sufficiently close to either of its bounds. From expression (28) the elements of u^* corresponding to the state variable bounds must be zero. Also those elements of $\partial\phi/\partial y$ corresponding to the decision variable bounds will be zero eliminating the last term of (32). Solving (32) for v^* and substituting into (31) will produce the following expression:

$$g(x^*)^T - g(y^*)^T \frac{\partial \psi}{\partial y^*}^{-1} \frac{\partial \psi}{\partial x^*} - u^{*T} \frac{\partial \phi}{\partial x^*} = 0 \tag{33}$$

Rearranging (33) we obtain

$$u^{*T} \frac{\partial \phi}{\partial x^*} = g(x^*)^T - g(y^*)^T \frac{\partial \psi}{\partial y}^{-1} \frac{\partial \psi}{\partial x} \tag{34}$$

We recognize the right-hand side of (34) to be $g_r(x)$. By examining the possible values of the left-hand side of (34), a candidate point x will be x^* if

$$g_r(x)_i \begin{cases} > 0 \text{ if } x_i = a_i \\ < 0 \text{ if } x_i = b_i \\ = 0 \text{ if } a_i \leq x_i \leq b_i \end{cases}$$

$$i = 1, 2, 3, \ldots, Q \tag{35}$$

Reduced Gradient—Algorithm

A flow chart of the Reduced Gradient Method to be described here is shown in Fig. 1. The basic steps involved are similar to those described by Abadie. The major differences described here occur in the calculation of the reduced gradient and the line search. Also it should be noted here that the present implementation of the method uses numerical techniques to obtain the required derivatives. In the results reported by Abadie, analytical derivatives were employed. However, obtaining analytical derivatives in many engineering applications can prove to be a tedious task and is often prone to error. In the following sections, implementation and computational considerations for each step in Fig. 1 will be discussed.

1 Specification of State and Decision Variables. In most

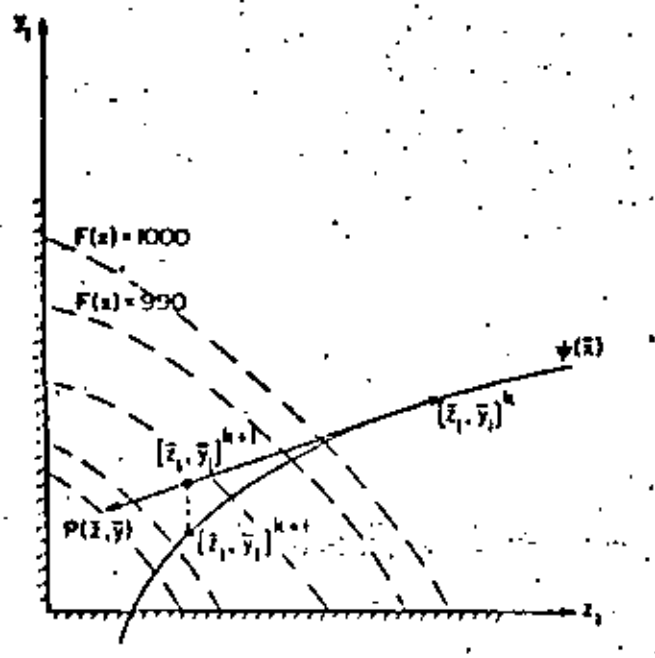


Fig. 2 Adjustment of state variable to obtain a feasible point during the linear search

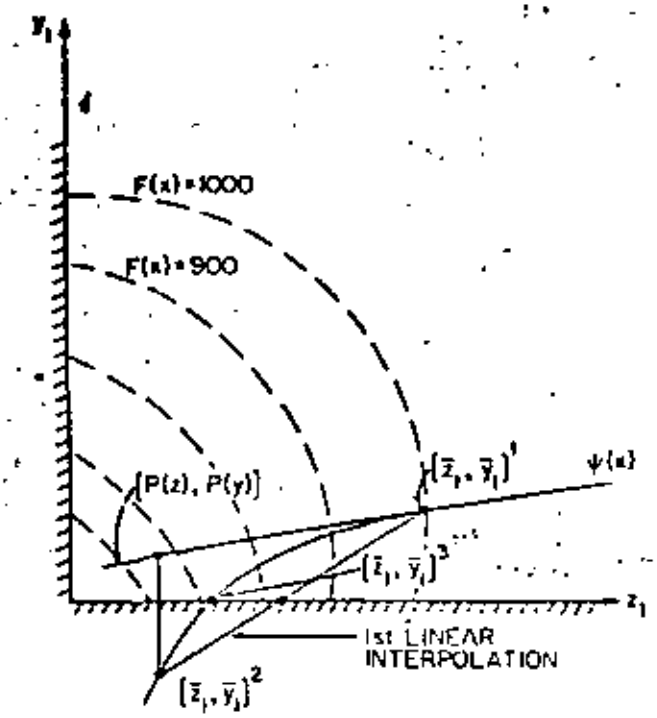


Fig. 3 Adjustment of state variable to locate a feasible point

and

$$x_m^{k+1} = x_m^k + \alpha(\Delta x_m) \qquad m = 1, 2, 3, \ldots, M \tag{13}$$

where α = step length parameter.

Because of nonlinearities arising in the constraint functions, the point $[x, y]^{k+1}$ is likely to be infeasible. Holding the decision variables x^{k+1} over final, the state variables y^{k+1} are adjusted to obtain a feasible point, $[x, y]^{k+1}$. This situation is shown in Fig. 2 for the case $M = 1$, $q = 1$. This is equivalent to the solution of M nonlinear equations $(g(x) = 0)$ in M unknowns (y). A number of numerical techniques exist to perform this task. Newton's method [16] has proven to be an efficient technique as well as convenient since the necessary partial derivatives have already been calculated.

At the completion of the adjustment procedure a new point $[x, y]^{k+1}$ has been determined and the following possible results must be considered:

(a) If all elements of y^{k+1} are within their specified bounds then $f(x)$ is evaluated at $[x, y]^{k+1}$ and the procedure for determining the minimum continues in the normal manner.

(b) If any element of y^{k+1} is not within its specified bounds, then $[x, y]^{k+1}$ is infeasible. Success (or linear interpolation is performed between the last feasible point $[x, y]^k$ and the point $[x, y]^{k+1}$ to determine the step length α which is the nearest bound encountered. This step length α should coincide with a single state variable equal to one of its bounds and all of the state variables within their specified bounds. Fig. 3 shows $[x_1, y_1]^3$ being out of bounds ($y_1 > D$). Using α moves it to the position $[x_1, y_1]^2$ which is bounded. Supplementary tests are then performed to determine whether the local constraint lies at the bound or at some point before it. If the minimum lies at the bound, the line search is terminated. If it lies before the bound, then the minimum has been bracketed and an improved value is found to be at the minimum.

(c) If the procedure fails to converge in a reasonable amount of time, the step length is reduced ($\alpha := \alpha/2$) and a new trial point generated.

6 Respecify State and Decision Variables. We saw in the previous step that it was possible to end a line search with a state variable equal to one of its specified bounds (second result tested previously). We start another iteration if the guidelines initially set forth for state variable selection must be satisfied. Those state variables that are bounded must be exchanged with decision variables that are unbounded and will not cause (x, y) to become singular.

Assume that y_1 is a bound state variable. Aboudi [16] suggests the following expression be *enumerated* to determine the appropriate decision variable to exchange with y_1.

$$\text{Min}\,|[D_x]_{ij}(b_i - x_i)|, \; |[D_y]_{ij}(y_j - a_j)| \qquad i = 1, 2, \ldots, q \tag{14}$$

where

$$D_x = \text{element of } \frac{\partial g}{\partial y}^{-1} \frac{\partial g}{\partial x} \tag{15}$$

Further Considerations and Recommendations

In the previous section the basic steps of a constrained gradient algorithm are described. In this section we would like to recommend some improvements and point out further modifications that can be made to the basic algorithm.

As with most nonlinear programming techniques, a large amount of effort is expended during the line search. Because of the reduced gradient's use of the state and decision variable dichotomy, much time is spent during the line search performing iterations of Newton's method to adjust the state variables. This time can be reduced through a modification of Newton's method given here:

$$y^{j+1} = y^j - \frac{\partial g}{\partial y}^{-1} g(x^k, y^j) \tag{16}$$

where

$\frac{\partial g}{\partial y}^{-1}$ = the initial inverse stored at the start to calculate the reduced gradient,

j = jth iteration of Newton's method

Although this modification does not preserve the convergence rate of the more lavish method, it does offer a considerable savings in time by avoiding successive reformulations of the Jacobian inverse.

The most stringent requirement of Newton's method is the need for a good initial approximation to the solution. Without it there is no guarantee of convergence. Aboudi has suggested linearization of

the constraints at the start to provide this initial approximation. This was described in the previous section. The following procedure offers a slight improvement:

1 From the initial starting point of the line search $[\bar{z}, \bar{y}]^0$, a step α_1 is taken along the search directions with the approximation of the new state variables taken as before

$$\bar{y}^1 = \bar{y}^0 + \alpha_1 \bar{t}^0(\bar{y})$$

2 Locate the feasible point $\bar{y}^1$ by Newton's method.

3 All other state variable approximations are generated by the following pattern step whenever Newton's method converges for the previous point

$$\bar{y}^k = \bar{y}^0 + \frac{\alpha_k}{\alpha_{k-1}}(\bar{y}^{k-1} - \bar{y}^0) \qquad (47)$$

where

$\bar{y}^{k-1}$ = the feasible point obtained at the previous step

α_{k-1} = the step length used for $[\bar{z}, \bar{y}]^{k-1}$.

This improvement in providing a starting point for Newton's

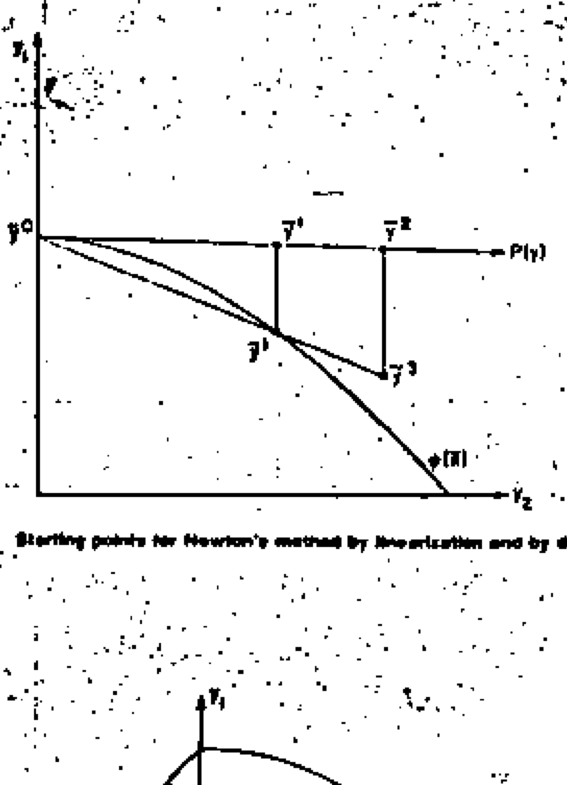


Fig. 4 Starting points for Newton's method by linearization and by direct move

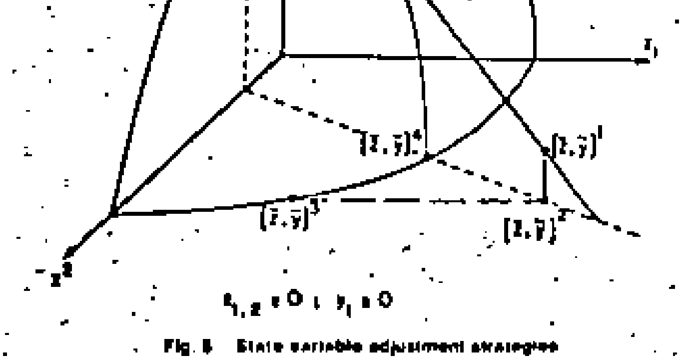


Fig. 5 State variable adjustment strategies

method decreases the number of Newton iterations required during the bounding phase, but is more readily exhibited during an iteration to a bound or a local minimum. Both approaches are pictured in Fig. 4.

The procedure outlined in (b) in the previous section on line searching differs from an approach that has been suggested by Abadie. In Fig. 5, we move from the point $[\bar{z}, \bar{y}]^1$ to the point $[\bar{z}, \bar{y}]^2$ as y_1 is adjusted. Since y_1 is nearly violating its lower bound, Abadie suggests the following scheme. Set y_1 to its lower bound and respecify it as a decision variable, choosing an appropriate decision variable to take its place. Now adjust the new state variable to maintain feasibility. This would result in the point $[\bar{z}, \bar{y}]^3$ shown in Fig. 5 with y_1 exchanged with z_1. Following the scheme given previously would result in the point $[\bar{z}, \bar{y}]^4$.

Both approaches seem to have advantages and it is unclear which approach is best. The main advantage of Abadie's line search is that it may more quickly locate those points where constraints intersect. However, if the modified Newton's method described previously is used, then a reformulation of the Jacobian inverse would be required every time a state variable was respecified. The approach described here leads to a less complicated algorithm and avoids frequent reformulation of the Jacobian inverse.

Results of Numerical Experiments With OPT

OPT is the name of our Reduced Gradient Fortran code, which implements the theory discussed in the previous sections. We report here testing of the program, which involved application of the code to each of the Eason and Fenton test problems. Eason and Fenton gave results for ten problems, three of which were originally given by Colville. Four of these ten problems are essentially unconstrained, in that the constraints are neither active at the solution nor encountered during the search. If constraints had been encountered during the search, the numerical results would have been included. Since they were not, the results indicate the performance of the unconstrained searching scheme only, which was not the subject of this study. Accordingly, these problems were not considered further. OPT has no difficulty in identifying interior optima, as is demonstrated in our solution of Eason and Fenton problem number 6. It is certainly worthy of note that the basic concept of the reduced gradient, the state-decision dichotomy, can be employed with any unconstrained searching algorithm. But this will be the subject of later results.

Table 1 gives the results obtained with OPT on Eason and Fenton problems number 1-3, 6, 7, and 9 [1]. In each case OPT terminated at a point which matched or exceeded the convergence criterion specified by Eason and Fenton, using standard program parameters. Listed is the number of objective and constraint function evaluations, and standard time needed to achieve solution.

OPT will terminate whenever the L_2 norm of the projected reduced gradient is less than 10^{-4}, or when the L_2 norm of the change in the design variables is less than 10^{-5}. The gradient condition caused termination in each case except problem 9. The line searches in OPT are conducted to an accuracy of 10^{-5}, and the tight constraints are satisfied to within 10^{-6} at every step of the search. Notice that OPT found solutions for all of the problems. Table 2 gives the number of

Table 1

Problem Number	N_f	N_c	Standard Time
1	44	171	0.0157
2	76	65	0.0085
3	14	154	0.0016
6	67	661	0.0017
7	77	147	0.0410
9	48	4,441	[illegible]

Table 2 Reduced gradient stages required for solution

Problem Number	Stages
1	6
2	2
3	3
6	4
7	3
9	10

Table 3 Relative ranking of optimization codes

N_s		T_N		$\bar{f}_a$		f_b	
6	OPT	1.	OPT	.013	OPT	.74	OPT
5.5	ADRANS	15.19	[illegible]	.30	[illegible]	8.9	[illegible]
	[illegible]	99.66	[illegible]	.50	[illegible]	40.7	[illegible]
	SIMPLX	194.66	[illegible]	.66	[illegible]	17.4	[illegible]
4	SEEK1	232.10	[illegible]	.71	MEMGRD	13.6	SIMPLX
4.5	[illegible]	256.60	[illegible]	.81	DAVID	40.0	[illegible]
3.5	MEMGRD	303.66	SIMPLX	1.30	SIMPLX	16.9	[illegible]
3	DAVID	479.84	[illegible]	1.66	[illegible]	16.0	MEMGRD

reduced gradient stages required to find each solution. Table 3 contains the rating schemes proposed by Eason and Fenton. N_s is the number of problems solved. The Eason and Fenton results were recomputed for the six problems considered here, and codes with an N_s value greater than or equal to three are included in Table 3. f_a and f_b are indicators of code efficiency defined by Eason and Fenton. T_N is the sum of the normalized times for each code on all problems. Our CDC 6500 system is rated at 51 s using Colville's standard timing program. Problems 6 and 7 have infeasible starting points. OPT handles this difficulty with the artificial variable approach suggested by Abadie. The artificial variable is used only to find an initial feasible starting point and plays no part in the remainder of the search.

The excellent performance demonstrated by OPT is the result of the ease with which the generalized reduced gradient algorithm identifies and follows active constraints. Additionally, the performance has been enhanced by careful attention to detail in the program implementation. Our experience indicates that even the most outstanding algorithm can perform poorly if the programming stage is improperly attended to.

Additional numerical results are reported in the thesis by Gabriele [17].

Conclusions

The following conclusions can be drawn from this work:

1 The state and decision variable dichotomy helps to reduce the constrained nonlinear programming problem to a simpler, more readily solved problem. By using the dichotomy the problem becomes, in simple terms, a search in the reduced unconstrained feasible region of the decision variables while maintaining feasibility through state variable adjustment.

2 The reduced gradient is a reliable and efficient tool for the solution of a wide variety of nonlinear programming problems, since OPT receives the top rating in each of the categories listed.

Acknowledgments

The authors are indebted to Honeywell Corporation for partial financial support of this project, and to the Purdue University Computing Center for their supply of computing power.

References

1 Eason, E. D., and Fenton, R. G., "A Comparison of Numerical Optimization Methods for Engineering Design," JOURNAL OF ENGINEERING FOR INDUSTRY, TRANS. ASME, Series B, Vol. 96, No. 1, Feb. 1974, pp. 196–200.

2 Colville, A. R., "A Comparative Study of Nonlinear Programming Codes," *Proceedings of the Princeton Symposium on Mathematical Programming*, Kuhn, H. W., ed., Princeton, N. J., 1970, pp. 487–501.

3 Fiacco, A. V., and McCormick, G. P., *Nonlinear Programming*, Wiley, New York, 1968.

4 Griffith, R. E., and Stewart, R. A., "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems," *Management Science*, Vol. 7, 1961, pp. 379–392.

5 Fox, R. L., *Optimization Methods for Engineering Design*, Addison-Wesley, 1971.

6 Himmelblau, D. M., *Applied Nonlinear Programming*, McGraw-Hill, New York, 1972.

7 Wolfe, P., "Methods for Linear Constraints," *Nonlinear Programming*, Abadie, J., ed., North Holland, Amsterdam, 1967, pp. 99–131.

8 Wolfe, P., "Methods of Nonlinear Programming," *Recent Advances in Mathematical Programming*, Graves, R. L., and Wolfe, P., eds., McGraw-Hill, New York, 1963, pp. 76–77.

9 Abadie, J., and Carpentier, J., "Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints," *Optimization*, Fletcher, R., ed., Academic Press, 1969, p. 37.

10 Wilde, D., and Beightler, C., *Foundations of Optimization*, Prentice-Hall, Englewood Cliffs, N. J., 1967, Chapters 2, 3.

11 Kuhn, W. W., and Tucker, A. W., "Nonlinear Programming," *Proceedings, 2nd Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, Berkeley, 1951, pp. 481–492.

12 Fletcher, R., and Reeves, C. M., "Function Minimization by Conjugate Gradients," *Computer Journal*, Vol. 6, No. 2, 1963, pp. 163–168.

13 Davidon, W. C., "Variable Metric Method for Minimization," Argonne National Laboratory, ANL-5990 Rev., University of Chicago, 1959.

14 Ragsdell, K. M., "A Survey of Some Useful Optimization Methods," *Proceedings of the Design Engineering Technical Conference*, New York, 1974, pp. 120–135.

15 Ragsdell, K. M., Root, R. R., and Gabriele, G. A., "Newton's Method: A Classical Technique of Contemporary Value," *Proceedings of the Design Engineering Technical Conference*, New York, 1974, pp. 137–145.

16 Abadie, J., "Application of the GRG Algorithm to Optimal Control Problems," *Integer and Nonlinear Programming*, Abadie, J., ed., North Holland, Amsterdam, 1970, p. 191.

17 Gabriele, G. A., "Application of the Reduced Gradient Method to Optimal Engineering Design," Master's thesis, School of Mechanical Engineering, Purdue University, Dec. 1975.

Printed in U.S.A.

Reprinted from JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS Vol. 4, No. 4, October 1969
Printed in Belgium

Sequential Gradient-Restoration Algorithm for the Minimization of Constrained Functions—Ordinary and Conjugate Gradient Versions[1]

A. MIELE,[2] H. Y. HUANG,[3] AND J. C. HEIDEMAN[4]

Abstract. The problem of minimizing a function $f(x)$ subject to the constraint $\varphi(x) = 0$ is considered. Here, f is a scalar, x an n-vector, and φ a q-vector. A *sequential algorithm* is presented, composed of the alternate succession of gradient phases and restoration phases.

In the *gradient phase*, a nominal point x satisfying the constraint is assumed; a displacement Δx leading from point x to a varied point y is determined such that the value of the function is reduced. The determination of the displacement Δx incorporates information at only point x for the *ordinary gradient* version of the method (Part 1) and information at both points x and $\bar{x}$ for the *conjugate gradient* version of the method (Part 2).

In the *restoration phase*, a nominal point y not satisfying the constraint is assumed; a displacement Δy leading from point y to a varied point ξ is determined such that the constraint is restored to a prescribed degree of accuracy. The restoration is done by requiring the least-square change of the coordinates.

If the stepsize α of the gradient phase is of $O(\epsilon)$, then $\Delta x = O(\epsilon)$ and $\Delta y = O(\epsilon^2)$. For ϵ sufficiently small, the restoration phase preserves the descent property of the gradient phase: the function f decreases between any two successive restoration phases.

[1] Paper received February 3, 1969. This research, supported by the NASA Manned Spacecraft Center, Grant No. NGR-44-006-038, and by the Office of Scientific Research, Office of Aerospace Research, United States Air Force, Grant No. AF-AFOSR-828-67, is a continuation of the investigations reported in Refs. 1 and 2.

[2] Professor of Astronautics and Director of the Aero-Astronautics Group, Department of Mechanical and Aerospace Engineering and Materials Science, Rice University, Houston, Texas.

[3] Assistant Professor of Astronautics, Department of Mechanical and Aerospace Engineering and Materials Science, Rice University, Houston, Texas.

[4] Graduate Student in Aero-Astronautics, Department of Mechanical and Aerospace Engineering and Materials Science, Rice University, Houston, Texas.

The ordinary gradient version of the algorithm exhibits asymptotic convergence, but not quadratic convergence. On the other hand, the conjugate gradient version exhibits quadratic convergence in the neighborhood of the minimum point. In particular, for a quadratic function subject to a linear constraint, the minimum point is obtained in no more than $n - q$ iterations.

1. Introduction

In recent years, considerable attention has been given to the iterative algorithms for minimizing a function $f(x)$ subject to the constraint $\varphi(x) = 0$, where f is a scalar, x an n-vector, and φ a q-vector. With reference to the gradient method, one possible approach is that of the penalty functions. The advantage of this approach is that the constrained minimal problem is replaced by a mathematically simpler, unconstrained minimal problem. The disadvantages are these: no clear-cut method exists for choosing the penalty constants; the algorithm must be repeated several times for increasing values of the penalty constants; the values of the function between iterations are not comparable, since the constraints are not satisfied; and, even when the algorithm is terminated, the constraints are generally not satisfied.

In this paper, we present a *sequential algorithm* constructed in such a way that the values of the function $f(x)$ between iterations are comparable. The algorithm is composed of the alternate succession of gradient phases and restoration phases (Fig. 1). In the *gradient phase*, a nominal point x satisfying the constraint is assumed; then, a displacement dx leading from point x to a varied point y is determined by minimizing the first variation $\delta f(x)$ subject to the linearized constraint $\delta\varphi(x) = 0$ and a quadratic constraint on dx. Due to the fact that the constraint is accounted for only to first order, the varied point y may be such that $\varphi(y) \neq 0$. This being the case, a restoration phase is needed prior to starting the next gradient phase. In this *restoration phase*, a nominal point y not satisfying the constraint is assumed; then, a displacement dy leading from point y to a varied point $\bar{x}$ is determined by minimizing the distance $dy^T dy$ subject to the linearized constraint $k\varphi(y) + \delta\varphi(y) = 0$, where $0 < k \leq 1$. If a single restoration cycle fails to produce the required degree of accuracy in the constraint, several cycles must be employed, as shown in Fig. 2. After the restoration phase is finished, the iteration is completed, and the next iteration is started using $\bar{x}$ as the nominal point x.

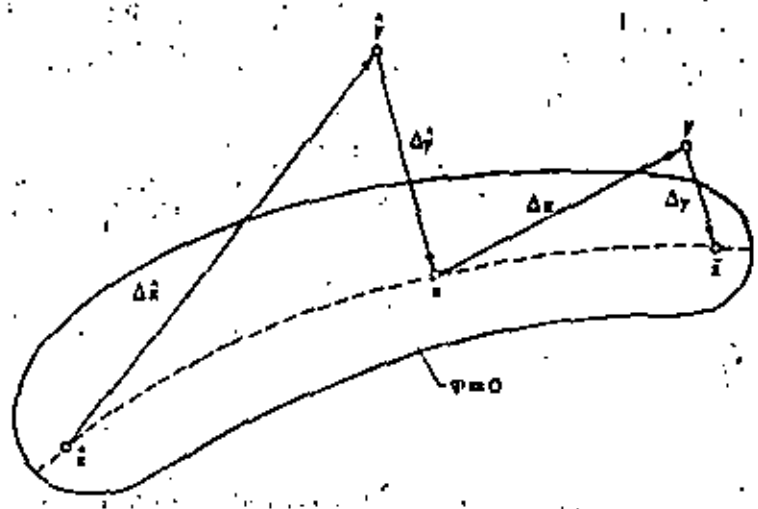


Fig. 1

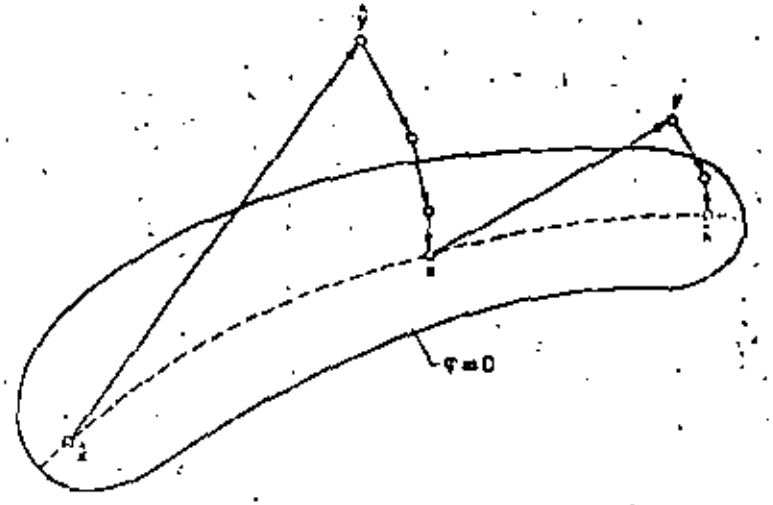


Fig. 2

Two versions of the method are presented: (a) the *sequential ordinary gradient-restoration algorithm*, in which the gradient phase uses information at point x only, and (b) the *sequential conjugate gradient-restoration algorithm*, in which the gradient phase uses information at both points x and $\hat{x}$, where $\hat{x}$ is the point preceding x. Method (a) is given in Part 1; Method (b) is given in Part 2; then, in Part 3, several numerical examples are presented.

2. Statement of the Problem

We consider the problem of minimizing the function

$$f = f(x) \tag{1}$$

subject to the constraint

$$\varphi(x) = 0 \tag{2}$$

In the above equation, f is a scalar, x an n-vector, and φ a q-vector,[6] where $q < n$. It is assumed that the first and second partial derivatives of the function f and the constraint φ with respect to x exist and are continuous; it is also assumed that the constrained minimum exists.

PART 1: SEQUENTIAL ORDINARY GRADIENT-RESTORATION ALGORITHM

In this part, we present the ordinary gradient version of the sequential gradient-restoration algorithm. The gradient phase is treated in Section 3 and the restoration phase in Section 4. In Section 5, an order-of-magnitude analysis is presented. Finally, in Section 6, the sequential ordinary gradient-restoration algorithm is summarized.

3. Gradient Phase

Consider a displacement Δx leading from a nominal point x to a varied point y such that

$$y = x + \Delta x \tag{3}$$

[6] All the vectors in this paper are column vectors.

Assume that the nominal point x satisfies (2) exactly and that the varied point y satisfies (2) to first order. The first-order change of the function (1) is given by

$$\delta f(x) = f_x^T(x)\,\Delta x \tag{4}$$

where $f_x(x)$ is the gradient of the scalar function f with respect to the vector x and the symbol T denotes the transpose of a matrix. In turn, the first-order change of the constraint (2) is represented as

$$\delta\varphi(x) = \varphi_x^T(x)\,\Delta x = 0 \tag{5}$$

where $\varphi_x(x)$, an $n \times q$ matrix, denotes the gradient of the vectorial function φ with respect to the vector x.

Next, consider the following quadratic constraint on the displacement Δx:

$$K = \Delta x^T \Delta x \tag{6}$$

where K is a constant. With this understanding, we formulate the following problem: Find the variation Δx which minimizes (4) subject to (5) (6).

3.1. Displacement Δx. Standard methods of the theory of maxima and minima show that the *fundamental function* of this problem is the scalar function

$$\Omega = f_x^T(x)\,\Delta x + \lambda^T\varphi_x^T(x)\,\Delta x + (1/2\alpha)\,\Delta x^T\,\Delta x \tag{7}$$

where $1/2\alpha$ is a scalar Lagrange multiplier and λ a q-vector Lagrange multiplier. If one introduces the *augmented function*

$$F(x, \lambda) = f(x) + \lambda^T\varphi(x) \tag{8}$$

and observes that

$$F_x(x, \lambda) = f_x(x) + \varphi_x(x)\,\lambda \tag{9}$$

the fundamental function (7) becomes

$$\Omega = F_x^T(x, \lambda)\,\Delta x + (1/2\alpha)\,\Delta x^T\,\Delta x \tag{10}$$

In Eqs. (9)-(10), the symbol $F_x(x, \lambda)$ denotes the gradient of the augmented function F with respect to the vector x. The optimal displacement Δx satisfies the relation

$$\Omega_{\Delta x} = 0 \tag{11}$$

where $\Omega_{\Delta x}$ denotes the gradient of the fundamental function Ω with respect to the vector Δx. The explicit form of (11) is the following:

$$\Delta x = -\alpha F_x(x, \lambda) \tag{12}$$

and shows that the displacement vector Δx has the same direction as the gradient of the augmented function F. Note that (12) determines Δx providing λ, α are specified.

3.2. Relation Between K and α. As Eq. (12) shows, the displacement Δx is proportional to α, the *stepsize* of the gradient phase. Upon substituting (12) into (6), we see that

$$K = \alpha^2 F_x^T(x, \lambda) F_x(x, \lambda) \tag{13}$$

Therefore, a correspondence exists between the values of the constant K and the values of the stepsize α. This being the case, one can bypass prescribing K and reason directly on α, as in the considerations which follow.

3.3. Determination of λ. If Eqs. (5), (9), (12) are combined, we obtain the relation

$$\varphi_x^T(x) f_x(x) + \varphi_x^T(x) \varphi_x(x) \lambda = 0 \tag{14}$$

Since the vector $f_x(x)$ and the matrix $\varphi_x(x)$ are known at the nominal point x, Eq. (14) supplies the multiplier λ; this is precisely the value which guarantees satisfaction of the constraint (2) to first order.

3.4. Descent Property of the Gradient Phase. The first variation of the augmented function is given by

$$\delta F(x, \lambda) = F_x^T(x, \lambda)\, \Delta x \tag{15}$$

which, in the light of (12), can be written as

$$\delta F(x, \lambda) = -\alpha F_x^T(x, \lambda) F_x(x, \lambda) \tag{16}$$

Equation (16) shows that, for $\alpha > 0$, the first variation of the augmented function is negative. Therefore, if α is sufficiently small, the augmented function F decreases during the gradient phase.

Alternatively, the first variation of $F(x, \lambda)$ can be written as

$$\delta F(x, \lambda) = \delta f(x) + \lambda^T \delta \varphi(x) \tag{17}$$

Because of (5), Eq. (17) reduces to the form

$$\delta f(x) = \delta F(x, \lambda) \tag{18}$$

which states that the functions $f(x)$ and $F(x, \lambda)$ behave identically, to first order. Therefore, the descent property also holds for the function f.

3.5. Stepsize. The next step is to assign a value to the stepsize α. If Eqs. (3) and (12) are combined, the position vector at the end of the gradient phase becomes

$$y = x - \alpha F_x(x, \lambda) \tag{19}$$

Since λ is known through Eq. (14), Eq. (19) defines a one-parameter family of points y for which the augmented function F takes the form

$$F(y, \lambda) = F(x - \alpha F_x(x, \lambda), \lambda) = \Psi(\alpha) \tag{20}$$

The greatest decrease of the function $\Psi(\alpha)$ occurs if the parameter α satisfies the following necessary condition:

$$\Psi_\alpha(\alpha) = 0 \tag{21}$$

After observing that

$$\Psi_\alpha(\alpha) = -F_y^T(y, \lambda) F_x(x, \lambda) \tag{22}$$

we see that Eq. (21) can be written as

$$F_x^T(y, \lambda) F_x(x, \lambda) = 0 \tag{23}$$

showing that the gradient of the augmented function at point y is orthogonal to the gradient at point x.

To obtain satisfaction of (21) or (23), some one-dimensional search method must be employed. In particular, *cubic interpolation* (it employs first derivatives only) and *quasilinearization* (it employs both first and second derivatives) are powerful methods (Refs. 1 and 3). These methods are to be employed iteratively until Eq. (21) is satisfied to a desired degree of accuracy, that is, until

$$\Psi_\alpha^2(\alpha) \leq \theta_1 \tag{24}$$

where θ_1 is a small number. In practice, θ_1 can be fixed or one may choose

$$\theta_1 = \epsilon_1 \Psi_\alpha^2(0) \tag{25}$$

where ϵ_1 is a small number.

4. Restoration Phase

At the end of the gradient phase, the point y is known. If the constraint is linear, the relation $\varphi(y) = 0$ holds. On the other hand, if the constraint is nonlinear, $\varphi(y) \neq 0$, which means that some degree of dissatisfaction exists. Therefore, a restoration phase is needed prior to starting the next gradient phase. Specifically, one has to apply a small variation Δy to y to generate a new position vector

$$\bar{x} = y + \Delta y \tag{26}$$

such that $\varphi(\bar{x}) = 0$. While there are infinite ways to perform the restoration, the most logical is that developed in Ref. 4: the constraint is restored to a prescribed degree of accuracy with the least-square change of the position vector.

If quasilinearization is employed, Eq. (2) is approximated by

$$\varphi(y) + \varphi_x^T(y)\,\Delta y = 0 \tag{27}$$

In order to prevent the variation Δy from becoming too large, we imbed Eq. (27) into the one-parameter family of equations

$$k\varphi(y) + \varphi_x^T(y)\,\Delta y = 0 \tag{28}$$

where

$$0 \leqslant k \leqslant 1 \tag{29}$$

denotes a scaling factor.

In the light of previous discussion, we seek the minimum of the function

$$J = \tfrac{1}{2}\,\Delta y^T\,\Delta y \tag{30}$$

subject to the linearized constraint (28). Standard methods of the theory of maxima and minima show that the *fundamental function* of this problem is given by

$$\omega = \tfrac{1}{2}\,\Delta y^T\,\Delta y + \sigma^T[k\varphi(y) + \varphi_x^T(y)\,\Delta y] \tag{31}$$

where σ, a q-vector, denotes an undetermined, constant Lagrange multiplier. The optimal change Δy satisfies the relation

$$\omega_{\Delta y} = 0 \tag{32}$$

where $\omega_{\Delta y}$ denotes the gradient of the fundamental function ω with respect to the vector Δy. The explicit form of (32) is the following:

$$\Delta y = -\varphi_x(y)\,\sigma \tag{33}$$

The Lagrange multiplier σ is obtained by combining (28) and (33) to eliminate Δy. This yields the relation

$$k\varphi(y) - \varphi_x^T(y)\,\varphi_x(y)\,\sigma = 0 \tag{34}$$

For any given k in the range (29), Eq. (34) supplies the Lagrange multiplier vector σ. Once σ is known, the correction Δy is given by (33) and the corrected position vector $\bar{x}$ is given by (26). Of course, the restoration phase must be performed iteratively until a desired degree of accuracy is obtained, that is, until the inequality

$$P(\bar{x}) \leqslant \theta_2 \tag{35}$$

is satisfied, where θ_2 is a small number and the performance index

$$P(x) = \varphi^T(x)\,\varphi(x) \tag{36}$$

measures the error in the constraint.

4.1. Descent Property of the Performance Index. The first variation of the performance index $P(y)$ is given by

$$\delta P(y) = 2\varphi^T(y)\,\varphi_x^T(y)\,\Delta y \tag{37}$$

and, because of Eq. (28), reduces to

$$\delta P(y) = -2k\varphi^T(y)\,\varphi(y) \tag{38}$$

which, in turn, can be written as

$$\delta P(y) = -2kP(y) \tag{39}$$

Since $P(y) \geqslant 0$, Eq. (39) shows that the first variation of the performance index is negative for $k > 0$. Therefore, if k is sufficiently small, the decrease of the performance index is guaranteed. In practice, one can use $k = 1$. If this value of k does not result in a decrease in P, then k is successively bisected until P is decreased.

5. Order-of-Magnitude Analysis

The position vector $\bar{x}$ at the end of the restoration phase and the position vector x at the beginning of the gradient phase are related by

$$\bar{x} = x + \delta x + \Delta y \tag{40}$$

where Δx is the gradient displacement and Δy is the restoration displacement. From Eq. (12), we see that, if the stepsize α is of $O(\epsilon)$, the gradient displacement has the order

$$\Delta x = O(\epsilon) \tag{41}$$

We observe that, to second order,

$$\varphi(y) = \varphi(x) + \varphi_x^T(x)\,\Delta x + \tfrac{1}{2}\,\Delta x^T \varphi_{xx}(x)\,\Delta x \tag{42}$$

where $\varphi_{xx}(x)$ denotes the array of the second partial derivatives of φ. Since the nominal point x satisfies (2) exactly and the variation Δx satisfies (2) to first order, the first two terms on the right-hand side of (42) vanish. Therefore, in the light of (41)–(42), we conclude that

$$\varphi(y) = O(\epsilon^2) \tag{43}$$

Next, we turn our attention to Eqs. (33)–(34) and observe that, because of (43),

$$\Delta y = O(\epsilon^2) \tag{44}$$

In conclusion, if the gradient displacement is of $O(\epsilon)$, the restoration displacement is of $O(\epsilon^2)$. This guarantees that, for sufficiently small ϵ,

$$|\Delta y| \ll |\Delta x| \tag{45}$$

5.1. Descent Property of the Algorithm. Finally, we consider the points x and $\bar{x}$, both satisfying the constraint (2). To first order, the difference of the values of the function f at these points is given by

$$f(\bar{x}) - f(x) = f_x^T(x)(\Delta x + \Delta y) \tag{46}$$

On account of (45), the second term on the right-hand side of (46) can be neglected with respect to the first. Hence, Eq. (46) becomes

$$f(\bar{x}) - f(x) = -\alpha F_x^T(x, \lambda) F_x(x, \lambda) \tag{47}$$

Therefore, for ϵ sufficiently small, the restoration algorithm preserves the descent property of the gradient algorithm: the function f decreases between any two successive restoration phases.

6. Summary of the Algorithm

The algorithm presented in Part 1 consists of the alternate succession of gradient phases and restoration phases. A summary of the algorithm is given below.

6.1. Gradient Phase. For this phase, the algorithm is represented by

$$\begin{gathered} \lambda = -(\varphi_x^T(x)\,\varphi_x(x))^{-1}\,\varphi_x^T(x) f_x(x), \\ F_x(x, \lambda) = f_x(x) + \varphi_x(x)\,\lambda \\ \Delta x = -\alpha F_x(x, \lambda) \\ y = x + \Delta x \end{gathered} \tag{48}$$

The sequence of operations is as follows: (a) select a nominal point x such that $\varphi(x) = 0$; (b) at this point, determine the vector $f_x(x)$ and the matrix $\varphi_x(x)$; (c) compute the multiplier λ with (48-1) and the vector $F_x(x, \lambda)$ with (48-2); (d) determine the optimal stepsize α by a one-dimensional search in which either $\Psi(\alpha) = f(y)$ or $\Psi(\alpha) = F(y, \lambda)$ is minimized along the search direction $F_x(x, \lambda)$; the search is terminated when Ineq. (24) is satisfied; (e) compute the displacement Δx with (48-3) and the varied point y with (48-4).

6.2. Restoration Phase. For this phase, the algorithm is represented by

$$\begin{gathered} \alpha = k[\varphi_x^T(y)\,\varphi_x(y)]^{-1}\,\varphi(y) \\ \Delta y = -\varphi_x(y)\,\alpha \\ \bar{y} = y + \Delta y \end{gathered} \tag{49}$$

The sequence of operations is as follows: (a) at point y, compute the vector $\varphi(y)$ and the matrix $\varphi_x(y)$; (b) assuming $k = 1$, determine the multiplier α with (49-1), the displacement Δy with (49-2), and the varied point $\bar{y}$ with (49-3); (c) if $P(\bar{y}) < P(y)$, the scaling factor $k = 1$ is acceptable; if $P(\bar{y}) > P(y)$, the previous value of k must be replaced by some smaller value in the range (29) until the condition $P(\bar{y}) < P(y)$ is met; this can be achieved through successive bisections of k; (d) return to step (a) and repeat the restoration algorithm using $\bar{y}$ as the starting point y for the subsequent iteration; (e) terminate the restoration algorithm when the stopping condition (35) is satisfied; (f) once the restoration algorithm is completed, verify the inequality

$$f(\bar{y}) < f(x) \tag{50}$$

If Ineq. (50) is satisfied, start the next gradient phase. If Ineq. (50) is violated, return to the previous gradient phase and reduce the stepsize α until, after restoration, Ineq. (50) is satisfied.

6.3. Stopping Condition. The algorithm is terminated when

$$Q(\bar{x}, \bar{\lambda}) \leq \theta_s \tag{51}$$

where θ_s is a small number and

$$Q(x, \lambda) = F_x^T(x, \lambda) F_\lambda(x, \lambda) \tag{52}$$

measures the error in the optimal conditions.

PART 2: SEQUENTIAL CONJUGATE GRADIENT-RESTORATION ALGORITHM

In this part, we present the conjugate gradient version of the sequential gradient-restoration algorithm. The gradient phase is discussed in Section 7 in general; the case of a quadratic function subject to a linear constraint is given in Section 8; then, the practical method to be used for a nonquadratic function subject to a nonlinear constraint is given in Section 9. The treatment of the restoration phase is omitted, since it is covered in Section 4. In Section 10, an order-of-magnitude analysis is presented. Finally, in Section 11, the sequential conjugate gradient-restoration algorithm is summarized.

7. Gradient Phase: General Discussion

Consider a displacement δx leading from a nominal point x to a varied point y given by Eq. (3). Assume that the nominal point x satisfies (2) exactly and that the varied point y satisfies (2) to first order. The first-order change of the function (1) is given by Eq. (4). In turn, the first-order change of the constraint (2) is represented by Eq. (5). Next, consider the following quadratic constraint on the displacement δx:

$$K = (\delta x - \beta\, \Delta\bar{x})^T (\delta x - \beta\, \Delta\bar{x}) \tag{53}$$

where K and β are constants and $\Delta\bar{x}$ is the displacement of the previous gradient phase, that is, the displacement leading from the nominal point $\bar{x}$ to the varied point $\bar{y}$ (see Figs. 1 and 2). With this understanding, we formulate the following problem: Find the displacement δx which minimizes (4) subject to (5) and (53).

7.1. Displacement δx. Standard methods of the theory of maxima and minima show that the *fundamental function* of this problem is the scalar function

$$\Omega = f_x^T(x)\, \delta x + \lambda^T F_x^T(x)\, \delta x + (1/2\alpha)(\delta x - \beta\, \Delta\bar{x})^T(\delta x - \beta\, \Delta\bar{x}) \tag{54}$$

where $1/2\alpha$ is a scalar Lagrange multiplier and λ a q-vector Lagrange multiplier. If one introduces the *augmented function* (8) and its gradient (9), the fundamental function (54) becomes

$$\Omega = F_x^T(x, \lambda)\, \delta x + (1/2\alpha)(\delta x - \beta\, \Delta\bar{x})^T(\delta x - \beta\, \Delta\bar{x}) \tag{55}$$

The optimal displacement δx satisfies the relation (11), whose explicit form is the following:

$$\delta x = -\alpha F_x(x, \lambda) + \beta\, \Delta\bar{x} \tag{56}$$

If one defines the search direction p from

$$\delta x = -\alpha p \tag{57}$$

and observes that, for the previous iteration,

$$\Delta\bar{x} = -\bar{\alpha}\bar{p} \tag{58}$$

the following relation ensues from (56)-(58):

$$p = F_x(x, \lambda) + \gamma\bar{p} \tag{59}$$

where

$$\gamma = \beta\bar{\alpha}/\alpha \tag{60}$$

In conclusion, the displacement δx during the gradient phase is given by (57), with p governed by Eq. (59), where $\bar{p}$ is known from the previous iteration. Note that (57) and (59) determine δx providing λ, α, γ are specified.

7.2. Relation Between K and α. As Eq. (57) shows, the displacement δx is proportional to α, the *stepsize* of the gradient phase. Upon substituting (56) into (53), we obtain Eq. (13). Therefore, a correspondence exists between the values of the constant K and the values of the stepsize α. This being the case, one can bypass prescribing K and reason directly on α, as in the considerations which follow.

7.3. Determination of λ, α, γ. If Eqs. (5), (9), (57), (59) are combined, we obtain the relation

$$\varphi_x^T(x) f_x(x) + \varphi_x^T(x) \varphi_x(x) \lambda + \gamma \varphi_x^T(x) \hat{p} = 0 \tag{61}$$

which ensures satisfaction of the constraint (2) to first order. Equation (61) is a linear relation between λ and γ and admits the solution

$$\lambda = -[\varphi_x^T(x) \varphi_x(x)]^{-1} [\varphi_x^T(x) f_x(x) + \gamma \varphi_x^T(x) \hat{p}] \tag{62}$$

Therefore, the rate of change of λ with respect to γ is given by the vector equation

$$\lambda_\gamma = -[\varphi_x^T(x) \varphi_x(x)]^{-1} \varphi_x^T(x) \hat{p} \tag{63}$$

The next step is to assign values to α and γ. If Eqs. (3), (57), (59) are combined, the position vector at the end of the gradient phase becomes

$$y = x - \alpha F_x(x, \lambda) - \alpha \gamma \hat{p} \tag{64}$$

Since λ depends on γ through Eq. (62), Eq. (64) defines a two-parameter family of points y for which the augmented function F takes the form

$$F(y, \lambda) = F[x - \alpha F_x(x, \lambda) - \alpha \gamma \hat{p}, \lambda] = \Psi(\alpha, \gamma). \tag{65}$$

The greatest decrease of the function $\Psi(\alpha, \gamma)$ occurs if the parameters α, γ satisfy the following necessary conditions:

$$\Psi_\alpha(\alpha, \gamma) = 0, \qquad \Psi_\gamma(\alpha, \gamma) = 0 \tag{66}$$

After observing that

$$\begin{aligned} \Psi_\alpha(\alpha, \gamma) &= -F_x^T(y, \lambda) \hat{p} \\ \Psi_\gamma(\alpha, \gamma) &= -\alpha F_x^T(y, \lambda) \hat{p} - [\alpha F_x^T(y, \lambda) \varphi_x(x) - \varphi^T(y)] \lambda_\gamma \end{aligned} \tag{67}$$

we see that Eqs. (66) can be written as

$$F_x^T(y, \lambda) \hat{p} = 0, \qquad \alpha F_x^T(y, \lambda) \hat{p} + [\alpha F_x^T(y, \lambda) \varphi_x(x) - \varphi^T(y)] \lambda_\gamma = 0 \tag{68}$$

In the light of (59) and (62)–(64), Eqs. (68) constitute a system of two scalar equations in the unknowns α and γ. Once the stepsize α and the coefficient γ are known from (68), the multiplier λ follows from (62), the search direction $\hat{p}$ from (59), the displacement Δx from (57), and the position vector y from (3). Thus, the problem of determining λ, α, γ is solved in principle; computationally, however, further simplifications are needed to make the algorithm practical.

8. Gradient Phase: Quadratic Function, Linear Constraint

Now, consider the particular case of a quadratic function and a linear constraint given in the form

$$f(x) = a + b^T x + \tfrac{1}{2} x^T c x, \qquad \varphi(x) = d + e^T x \tag{69}$$

where a is a scalar, b is an n-vector, c an $n \times n$ symmetric matrix, d a q-vector, and e an $n \times q$ matrix. Here, all the coefficients are constant. The gradients of the functions f and φ become

$$f_x(x) = b + cx, \qquad \varphi_x(x) = e \tag{70}$$

Because of the linearity, the constraint is never violated during the gradient phase and, consequently,

$$\Delta y = 0, \qquad \bar{x} = x + \Delta x, \qquad \varphi_x^T(x) \hat{p} = 0 \tag{71}$$

This being the case, Eq. (62) supplies the following expression for the multiplier:

$$\lambda = -[\varphi_x^T(x) \varphi_x(x)]^{-1} \varphi_x^T(x) f_x(x) \tag{72}$$

which is now independent of γ, that is,

$$\lambda_\gamma = 0 \tag{73}$$

The relations (68) optimizing α and γ become

$$F_x^T(y, \lambda) \hat{p} = 0, \qquad F_x^T(y, \lambda) \hat{p} = 0 \tag{74}$$

showing that the gradient of the augmented function at point y is orthogonal to both the present and previous search directions. A mathematical consequence of (59) and (74) is that

$$F_x^T(y, \lambda) F_x(x, \lambda) = 0 \tag{75}$$

showing that the gradients at point y and point x are orthogonal. Furthermore, after laborious manipulations, Eqs. (74)–(75) lead to

$$F_x^T(\bar{x}, \bar{\lambda}) \hat{p} = 0, \qquad F_x^T(\bar{x}, \bar{\lambda}) \hat{p} = 0, \qquad F_x^T(\bar{x}, \bar{\lambda}) F_x(x, \lambda) = 0 \tag{76}$$

For a quadratic function subject to a linear constraint, the following relationship can be shown to hold:

$$F_x(y, \lambda) = F_x(x, \lambda) - \alpha c \hat{p} \tag{77}$$

Invoking (69)–(77), we see that (74-1) yields the following solution for the optimal stepsize:

$$\alpha = F_x^T(x, \lambda)\, \nabla_x(x, \lambda)/p^T c p \tag{78}$$

Furthermore, (74-2) leads to

$$p^T c \hat{p} = 0 \tag{79}$$

which states that the search directions p and $\hat{p}$ are conjugate with respect to the matrix c. In turn, (79) yields the following explicit solution for γ:

$$\gamma = F_x^T(x, \lambda)\, F_x(x, \lambda)/F_x^T(\hat{x}, \hat{\lambda})\, F_x(\hat{x}, \hat{\lambda}) \tag{80}$$

In conclusion, for a given nominal point x, the multiplier λ is supplied by (72), the coefficient γ by (80), the search direction p by (59), the optimum stepsize α by (78), the displacement Δx by (57), and the position vector y by (3).

8.1. Convergence Properties. For a quadratic function subject to a linear constraint, the following relations can be shown to hold *provided the first step of the algorithm is a gradient step*:

$$F_x^T(x, \lambda)\, F_x(x_v, \lambda_v) = 0, \qquad F_x^T(x, \lambda)\, p_v = 0, \qquad p^T c p_v = 0 \tag{81}$$

where x_v denotes any state preceding x. Equations (81) can be derived from (76) and (79) through mathematical induction. The first of Eqs. (81) states that the gradient at each point is orthogonal to the gradient at every previous point. The second of Eqs. (81) states that the gradient at each point is orthogonal to the search direction at every previous point. Finally, the third of Eqs. (81) states that the search direction at each point and the search direction at every previous point are conjugate with respect to the constant matrix c; this is why the algorithm is called the *conjugate-gradient algorithm*.

Laborious manipulations, omitted for the sake of brevity, show that the algorithm defined by (3), (57), (59) with λ, α, γ defined by (72), (78), (80) reduces the gradient $F_x(x, \lambda)$ to zero in no more than $n - q$ steps; therefore, the minimum of the function $f(x)$ subject to the constraint $\varphi(x) = 0$ is reached in no more than $n - q$ steps. If the function is unconstrained, that is, if $q = 0$, then the minimum of $f(x)$ is reached in no more than n steps (see Refs. 5 and 6).

9. Gradient Phase: Nonquadratic Function and/or Nonlinear Constraint

If the function $f(x)$ is nonquadratic and/or the constraint $\varphi(x)$ is nonlinear, the relations (72), (78), (80) defining λ, α, γ are not simultaneously valid. We observe that (72) and (80) involve first derivatives only, while (78) involves the second-derivative matrix c. This being the case, we choose to discard (78) and retain (72) and (80). Thus, we employ the algorithm

$$\begin{aligned} \lambda &= -[\nabla_x^T(x)\, \nabla_x(x)]^{-1} \nabla_x^T(x) f_x(x) \\ F_x(x, \lambda) &= f_x(x) + \nabla_x(x)\, \lambda \\ \gamma &= F_x^T(x, \lambda)\, F_x(x, \lambda)/F_x^T(\hat{x}, \hat{\lambda})\, F_x(\hat{x}, \hat{\lambda}) \\ p &= F_x(x, \lambda) + \gamma \hat{p} \\ \Delta x &= -\alpha p \\ y &= x + \Delta x \end{aligned} \tag{82}$$

which, for the unconstrained case, reduces to the well-known Fletcher–Reeves algorithm (see Refs. 3 and 7). For the constrained case, the justification of the expressions for the multiplier λ and the coefficient γ is presented in Section 10. Once the nominal point x is given, the multiplier λ is determined with (82-1), the gradient $F_x(x, \lambda)$ with (82-2), the coefficient γ with (82-3), and the search direction p with (82-4); for a given stepsize α, the displacement Δx is computed with (82-5) and the varied point y with (82-6). The determination of α is discussed in the following section.

9.1. Stepsize. If Eqs. (82-5) and (82-6) are combined, the position vector at the end of the gradient phase becomes

$$y = x - \alpha p \tag{83}$$

For a given nominal point x, the vector p is known through Eqs. (82-1)–(82-4); therefore, Eq. (83) defines a one-parameter family of points y for which the augmented function $F(y, \lambda)$ takes the form

$$F(y, \lambda) = F(x - \alpha p, \lambda) = \Psi(\alpha) \tag{84}$$

The greatest decrease in the function $\Psi(\alpha)$ occurs if the parameter α satisfies the following necessary condition:

$$\Psi_\alpha(\alpha) = 0 \tag{85}$$

After observing that

$$\Psi_x(\alpha) = -F_x^T(y, \lambda)\, p \tag{86}$$

we see that Eq. (85) can be written as

$$F_x^T(y, \lambda)\, p = 0 \tag{87}$$

showing that the gradient of the augmented function at point y is orthogonal to the search direction p.

To obtain satisfaction of (85) or (87), some one-dimensional search method must be employed. In particular, *cubic interpolation* (it employs first derivatives only) and *quasilinearization* (it employs both first and second derivatives) are powerful methods (Refs. 1–3). These methods are to be employed iteratively until Eq. (85) is satisfied to a desired degree of accuracy, that is, until Ineq. (24) is satisfied.

9.2. Starting the Algorithm. The algorithm (82) requires that the search direction $\hat{p}$ be known from the previous iteration. Since this is not the case for the first iteration, some assumption concerning $\gamma\hat{p}$ is needed in order to start the algorithm. To retain quadratic convergence, one must choose $\hat{p} = 0$ or $\gamma = 0$. Consequently, for the first step, the search direction (82–4) becomes

$$p = F_x(x, \lambda) \tag{88}$$

meaning that the first step is a pure gradient step.

9.3. Restarting the Algorithm. For a quadratic function subject to a linear constraint, the present algorithm converges to the exact minimum in no more than $n - q$ steps. For the general case, this suggests the idea of restarting the algorithm every $\Delta N = n - q$ or $\Delta N = n - q + 1$ steps.

10. Order-of-Magnitude Analysis

The sequential conjugate gradient-restoration algorithm is represented by Eqs. (82) and (49). While Eqs. (49) are exact, Eqs. (82) are an approximation to the true optimal conditions. In this section, an estimate of the error involved in the computation of the Lagrange multiplier λ is given; furthermore, a verification of the descent properties of the algorithm is presented.

10.1. Lagrange Multiplier. Here, we assume that, if ϵ is a small quantity,

$$\alpha = O(\epsilon) \tag{89}$$

for every gradient phase. Concerning the multiplier λ, we note that the exact equation (62) has been replaced by the approximate equation (82-1); therefore, the q-vector

$$R_1 = \gamma\varphi_x^T(x)\,\hat{p} \tag{90}$$

is representative of the order of magnitude of the error in λ. This is due to the fact that the terms $\varphi_x^T(x) f_x(x)$ and $\varphi_x^T(x)\varphi_x(x)$ in Eq. (62) can be regarded to be of $O(1)$.

If λ is computed with the approximate equation (82-1), the first-order change of the constraint (2) is not exactly zero and, because of Eqs. (82), is given by

$$\delta\varphi(x) = -\alpha\gamma\varphi_x^T(x)\,\hat{p} \tag{91}$$

If one observes that $\varphi(x) = 0$ and that, to first order,

$$\delta\varphi(x) = \varphi(y) - \varphi(x) = \varphi(y) \tag{92}$$

it follows that

$$\varphi(y) = -\alpha\gamma\varphi_x^T(x)\,\hat{p} \tag{93}$$

If Eqs. (49) and (93) are combined, the displacement Δy associated with the restoration phase can be written as

$$\Delta y = \alpha\gamma h\varphi_x(y)[\varphi_x^T(y)\varphi_x(y)]^{-1}\varphi_x^T(x)\,\hat{p} \tag{94}$$

To first order, the expansion of the gradient $\varphi_x(\bar{x})$ is given by

$$\varphi_x(\bar{x}) = \varphi_x(x) + \varphi_{xx}(x)(\Delta x + \Delta y) \tag{95}$$

where $\varphi_{xx}(x)$ denotes the array of the second partial derivatives of φ. If the transposes of both sides of (95) are postmultiplied by $\hat{p}$ and if Eqs. (82-5) and (94) are accounted for, one deduces that

$$\varphi_x^T(\bar{x})\,p = \varphi_x^T(x)\,p - \alpha p^T\varphi_{xx}(x)\,p + \alpha\gamma h p^T\varphi_{xx}(x)\,\varphi_x(y)[\varphi_x^T(y)\,\varphi_x(y)]^{-1}\varphi_x^T(x)\,\hat{p} \tag{96}$$

Because of (82-1) and (82-4), the first term on the right-hand side of (96) can be written as

$$\varphi_x^T(x)\,p = \gamma\varphi_x^T(x)\,\hat{p} \tag{97}$$

As a consequence, the third term on the right-hand side of (96) is negligible with respect to the first and Eq. (96) can be approximated by

$$\varphi_x^T(\bar{x})\,\bar{p} = \gamma\varphi_x^T(x)\,\bar{p} - \alpha\bar{p}^T\varphi_{xx}(x)\,\bar{p} \tag{98}$$

This is the key recurrence formula necessary to estimate the order of magnitude of the terms of the form $\varphi_x^T(x)\bar{p}$ or $\varphi_x^T(\bar{x})\bar{p}$. Since α is of $O(\epsilon)$, we see that, for the first step of the algorithm ($\gamma = 0$ or $\bar{p} = 0$),

$$\varphi_x^T(\bar{x})\,\bar{p} = O(\epsilon) \tag{99}$$

Therefore, for any subsequent step, an analogous relation holds. Thus, at any point x, we conclude that

$$\varphi_x^T(x)\,\bar{p} = O(\epsilon) \tag{100}$$

Since R_1 given by (90) is proportional to the left-hand side of (100), we see that

$$R_1 = O(\epsilon) \tag{101}$$

Therefore, the Lagrange multiplier λ computed with (82-1) is precise to $O(\epsilon)$.

10.2. Remark. Because of (89), (91), (93), (100), we conclude that

$$\delta_\Psi(x) = O(\epsilon^2), \qquad \varphi(y) = O(\epsilon^2) \tag{102}$$

Furthermore, from (82-5), (89), (94), (100), we see that

$$\Delta x = O(\epsilon), \qquad \Delta y = O(\epsilon^2) \tag{103}$$

10.3. Descent Property of the Gradient Phase. The first-order change of the function $F(x, \lambda)$ between points x and y is given by Eq. (15); in the light of (82), this can be written as

$$\delta F(x, \lambda) = -\alpha F_x^T(x, \lambda)\,F_x(x, \lambda) - R_1 \tag{104}$$

where R_1 is a scalar given by

$$R_1 = \alpha\gamma F_x^T(x, \lambda)\,\bar{p} \tag{105}$$

On account of (9), Eq. (105) can be rewritten as

$$R_1 = R_2 + R_3 \tag{106}$$

where

$$R_2 = \alpha\gamma[f_x(x) + \varphi_x(x)\,\bar{\lambda}]^T\bar{p}, \qquad R_3 = \alpha\gamma(\lambda - \bar{\lambda})^T\varphi_x^T(x)\,\bar{p} \tag{107}$$

We observe that, to first order,

$$f_x(\bar{y}) = f_x(\bar{y}) + f_{xx}(\bar{y})\,\Delta\bar{y}, \qquad \varphi_x(x) = \varphi_x(\bar{y}) + \varphi_{xx}(\bar{y})\,\Delta\bar{y} \tag{108}$$

and that

$$\lambda - \bar{\lambda} = \Phi_x^T(\bar{x})(\Delta\bar{x} + \Delta\bar{y}) \tag{109}$$

where $\Phi(x)$ denotes the right-hand side of (82-1). In the light of Eq. (87) applied to the previous iteration, Eqs. (107) become

$$R_2 = \alpha\gamma\,\Delta\bar{y}^T F_{xx}(\bar{y}, \bar{\lambda})\,\bar{p}, \qquad R_3 = \alpha\gamma(\Delta\bar{x} + \Delta\bar{y})^T\Phi_x(\bar{x})\,\varphi_x^T(x)\,\bar{p} \tag{110}$$

If (89), (100), (103) are recalled, we see that

$$R_2 = O(\epsilon^3), \qquad R_3 = O(\epsilon^3) \tag{111}$$

so that

$$R_1 = O(\epsilon^3) \tag{112}$$

Since the first term on the right-hand side of Eq. (104) is of $O(\epsilon)$, R_1 can be neglected and Eq. (104) can be approximated by

$$\delta F(x, \lambda) = -\alpha F_x^T(x, \lambda)\,F_x(x, \lambda) \tag{113}$$

This relationships shows that, if $\alpha > 0$, $\delta F(x, \lambda) < 0$. This is the descent property of the fundamental function during the gradient phase.

Because of definition (8), the relationship (17) can be established. Since $\delta F(x, \lambda)$ is of $O(\epsilon)$ and $\delta_\Psi(x)$ is of $O(\epsilon^2)$, Eq. (17) can be approximated by

$$\delta f(x) = \delta F(x, \lambda) \tag{114}$$

which states that the functions $f(x)$ and $F(x, \lambda)$ behave identically, to first order. This and (113) establish the descent property of the function $f(x)$ during the gradient phase.

10.4. Descent Property of the Algorithm. Finally, we consider points x and $\bar{x}$, both satisfying the constraint (2). To first order, the difference of the values of the function $f(x)$ at these points is given by

$$f(\bar{x}) - f(x) = f_x^T(x)(\Delta x + \Delta_\Psi) \tag{115}$$

On account of (103), the second term on the right-hand side of (115) can be neglected with respect to the first. Hence, Eq. (115) becomes

$$f(\bar{x}) - f(x) = -\alpha F_x^T(x, \lambda)\,F_x(x, \lambda) \tag{116}$$

Therefore, for ϵ sufficiently small, the restoration algorithm preserves the descent property of the gradient algorithm: the function f decreases between any two successive restoration phases.

10.5. Coefficient γ. In the present algorithm, expression (82-3) has been used for the coefficient γ. The main justification for (82-3) is that it produces quadratic convergence in the terminal stage of the algorithm. The additional justification is that the terms (90) and (105) containing γ are of $O(\epsilon)$ and $O(\epsilon^2)$, respectively. As a consequence, they do not seriously affect the computation of the multiplier λ and the descent property of the algorithm.

11. Summary of the Algorithm

The algorithm presented in Part 2 consists of the alternate succession of gradient phases and restoration phases. A summary of the algorithm is given below.

11.1. Gradient Phase. For this phase, the algorithm is represented by

$$\begin{aligned} \lambda &= -[\varphi_x^T(x)\,\varphi_x(x)]^{-1}\varphi_x^T(x)\,f_x(x) \\ F_x(x,\lambda) &= f_x(x) + \varphi_x(x)\,\lambda \\ \gamma &= F_x^T(x,\lambda)\,F_x(x,\lambda)\,F_x^T(\bar{x},\bar{\lambda})\,F_x(\bar{x},\bar{\lambda}) \\ p &= F_x(x,\lambda) + \gamma\bar{p} \\ \Delta x &= -\alpha p \\ y &= x + \Delta x \end{aligned} \tag{117}$$

The sequence of operations is as follows: (a) select a nominal point $\bar{x}$ such that $\varphi(\bar{x}) = 0$; (b) at this point, determine the vector $f_x(x)$ and the matrix $\varphi_x(x)$; (c) compute the multiplier λ with (117-1), the vector $F_x(x,\lambda)$ with (117-2), the coefficient γ with (117-3), and the search direction p with (117-4), where $\bar{p}$ is known from the previous iteration; (d) determine the optimal stepsize α by a one-dimensional search in which either $\Psi(\alpha) = f(y)$ or $\Psi(\alpha) = F(y,\lambda)$ is minimized along the search direction p; the search is terminated when Ineq. (24) is satisfied; (e) compute the displacement Δx with (117-5) and the varied point y with (117-6).

11.2. Restoration Phase. For this phase, the algorithm is represented by

$$\begin{aligned} \sigma &= h[\varphi_x^T(y)\,\varphi_x(y)]^{-1}\varphi(y) \\ \Delta y &= -\varphi_x(y)\,\sigma \\ \bar{x} &= y + \Delta y \end{aligned} \tag{118}$$

The sequence of operations is as follows: (a) at point y, compute the vector $\varphi(y)$ and the matrix $\varphi_x(y)$; (b) assuming $k = 1$, determine the multiplier σ with (118-1), the displacement Δy with (118-2), and the varied point $\bar{x}$ with (118-3); (c) if $P(\bar{x}) < P(y)$, the scaling factor $k = 1$ is acceptable; if $P(\bar{x}) > P(y)$, the previous value of k must be replaced by some smaller value in the range (29) until the condition $P(\bar{x}) < P(y)$ is met; this can be achieved through successive bisections of k; (d) return to step (a) and repeat the restoration algorithm using $\bar{x}$ as the starting point y for the subsequent iteration; (e) terminate the restoration algorithm when the stopping condition (35) is satisfied; (f) once the restoration algorithm is completed, verify the inequality

$$f(\bar{x}) < f(x) \tag{119}$$

If Ineq. (119) is satisfied, start the next gradient phase. If Ineq. (119) is violated, return to the previous gradient phase and reduce the stepsize α until, after restoration, Ineq. (119) is satisfied.

11.3. Starting Condition. At the start of the algorithm, no information pertaining to the previous iteration is available; hence, we set $\bar{p} = 0$ or $\gamma = 0$. This means that the first step is a pure gradient step.

11.4. Restarting Condition. The algorithm must be restarted (a) when the optimal stepsize α cannot be employed due to violation of Ineq. (119) and (b) at the end of every $\Delta N = n - q$ or $\Delta N = n - q + 1$ iterations. The restarting is performed by setting $\bar{p} = 0$ or $\gamma = 0$.

11.5. Stopping Condition. The algorithm is terminated when Ineq. (51) is satisfied.

PART 3: NUMERICAL EXAMPLES

In order to illustrate the theory, several numerical examples are developed using a Burroughs B-5500 computer and double-precision arithmetic. Concerning the *gradient phase*, the one-dimensional search is performed so as

to minimize either $\Psi(\alpha) = f(y)$ or $\Psi(\alpha) = F(y, \lambda)$ with respect to α along the search direction. Quasilinearization is used; the following stopping condition is employed:

$$\Psi_{,\alpha}^{2}(\alpha)/\Psi_{,\alpha}^{2}(0) \leqslant 10^{-8} \tag{120}$$

corresponding to $\varepsilon_1 = 10^{-8}$ in (25). Concerning the *restoration phase*, the restoration algorithm is employed iteratively until the error in the constraint satisfies the inequality

$$I(\bar{x}) \leqslant 10^{-20} \tag{121}$$

corresponding to $\theta_3 = 10^{-20}$ in (35). Finally, the *sequential gradient-restoration algorithm* is terminated when the error in the optimal conditions satisfies the inequality

$$Q(\bar{x}, \bar{\lambda}) \leqslant 10^{-12} \tag{122}$$

corresponding to $\theta_2 = 10^{-12}$ in (51).

In the following sections, two groups of numerical examples are presented: (a) examples pertaining to quadratic function and linear constraint and (b) examples pertaining to nonquadratic function and/or nonlinear constraint. Both the ordinary gradient and the conjugate gradient versions of the algorithm are used. The following terminology is adopted: N is the iteration number (each iteration includes a gradient phase and a restoration phase), N_R is the number of restoration cycles per iteration, N_* is the number of iterations for convergence, and ΔN is the number of iterations between successive restarting points. For simplicity, the symbols employed throughout Part 3 are scalar.

12. Examplest Quadratic Function, Linear Constraint

The first group of examples deals with functions of the form (69-1) subject to a constraint of the form (69-2). For these functions, the following properties hold: (a) no restoration is needed, that is, $N_R = 0$; (b) the search performed by quasilinearization yields the optimal value of α in one step; (c) the value of α computed by minimizing $\Psi = f$ is the same as that computed by minimizing $\Psi = F$; and (d) convergence to the minimum occurs in no more than $n - q$ iterations in the conjugate gradient version of the algorithm.

Example 12.1. We consider the problem of minimizing the function

$$f = (x + y)^2 + (y + z)^2 \tag{123}$$

subject to the constraint

$$x + 2y + 3z - 1 = 0 \tag{124}$$

Note that $n = 3$, $q = 1$, so that $n - q = 2$. This function admits the relative minimum $f = 0$ at the point defined by

$$x = \tfrac{1}{2}, \quad y = -\tfrac{1}{2}, \quad z = \tfrac{1}{2} \tag{125}$$

The nominal point chosen for starting the algorithm is the point of coordinates

$$x = -4, \quad y = 1, \quad z = 1 \tag{126}$$

consistent with (124). Convergence is achieved in $N_* = 19$ iterations with the ordinary gradient version of the algorithm and $N_* = 2$ iterations with the conjugate gradient version. For the latter, Table 1 shows the detailed results.

Table 1 ($n - q = 2$)

N	N_R	x	y	z	f	Q
0	—	−4.0000	1.0000	1.0000	0.13×10^{2}	0.55×10^{2}
1	0	−1.6439	1.8759	−0.3686	0.33×10^{1}	0.20×10^{1}
2	0	0.5000	−0.5000	0.5000	0.93×10^{-16}	0.50×10^{-11}

Example 12.2. We consider the problem of minimizing the function

$$f = (x - 1)^2 + (y - z)^2 + (u - v)^2 \tag{127}$$

subject to the constraints

$$x + y + z + u + v - 5 = 0, \qquad x - 2(u + v) + 3 = 0 \tag{128}$$

Note that $n = 5$, $q = 2$, so that $n - q = 3$. This function admits the relative minimum $f = 0$ at the point defined by

$$x = 1, \quad y = 2, \quad z = 1, \quad u = 1, \quad v = 1 \tag{129}$$

The nominal point chosen for starting the algorithm is the point of coordinates

$$x = 3, \quad y = 5, \quad z = -3, \quad u = 2, \quad v = -2 \tag{130}$$

consistent with (128). Convergence is achieved in $N_s = 17$ iterations with the ordinary gradient version of the algorithm and $N_s = 3$ iterations with the conjugate gradient version. For the latter, Table 2 shows the detailed results.

Table 2 ($n - q = 3$)

N	N_s	x	y	z	u	w	f	Q
0	—	3.0000	5.0000	−3.0000	2.0000	−2.0000	0.84×10^{1}	0.62×10^{2}
1	0	1.8499	0.6651	0.6565	0.7709	1.0373	0.78×10^{0}	0.26×10^{1}
2	0	1.1904	0.8369	0.9417	1.1059	0.8549	0.10×10^{0}	0.63×10^{0}
3	0	1.0000	1.0000	1.0000	1.0000	1.0000	0.12×10^{-11}	0.69×10^{-10}

Example 12.3. We consider the problem of minimizing the function

$$f = (x - y)^4 + (y + z - 2)^4 + (u - 1)^4 + (w - 1)^4 \tag{131}$$

subject to the constraints

$$x + 3y - 4 = 0, \quad x + u - 2w = 0, \quad y - w = 0 \tag{132}$$

Note that $n = 5$, $q = 3$, so that $n - q = 2$. This function admits the relative minimum $f = 0$ at the point defined by

$$x = 1, \quad y = 1, \quad z = 1, \quad u = 1, \quad w = 1 \tag{133}$$

The nominal point chosen for starting the algorithm is the point of coordinates

$$x = \tfrac{1}{2}, \quad y = \tfrac{1}{2}, \quad z = 2, \quad u = -1, \quad w = \tfrac{1}{2} \tag{134}$$

consistent with (132). Convergence is achieved in $N_s = 13$ iterations with the ordinary gradient version of the algorithm and $N_s = 2$ iterations with the conjugate gradient version. For the latter, Table 3 shows the detailed results.

Table 3 ($n - q = 2$)

N	N_s	x	y	z	u	w	f	Q
0	—	2.5000	0.5000	2.0000	−1.0000	0.5000	0.85×10^{1}	0.40×10^{2}
1	0	0.9615	1.0962	1.5769	0.5288	1.0481	0.73×10^{0}	0.37×10^{1}
2	0	1.0000	1.0000	1.0000	1.0000	1.0000	0.70×10^{-11}	0.30×10^{-10}

13. Examples: Nonquadratic Function and/or Nonlinear Constraint

The second group of examples deals with general functions and general constraints. Concerning the search, the value of α computed by minimizing $\Psi = f$ is different from that computed by minimizing $\Psi = F$. For the conjugate gradient version, the algorithm is restarted every ΔN iterations. Note that $\Delta N = 1$ corresponds to the ordinary gradient version.

Example 13.1. We consider the problem of minimizing the function

$$f = (x - y)^4 + (y - z)^4 \tag{135}$$

subject to the constraint

$$x(1 + y^2) + z^4 - 3 = 0 \tag{136}$$

Note that $n = 3$, $q = 1$, so that $n - q = 2$. This function admits the relative minimum $f = 0$ at the point defined by

$$x = 1, \quad y = 1, \quad z = 1 \tag{137}$$

The nominal point chosen for starting the algorithm is the point of coordinates

$$x = -13/5, \quad y = 2, \quad z = 2 \tag{138}$$

consistent with (136). Table 4 shows N_s for several values of ΔN and both $\Psi = f$ and $\Psi = F$. The detailed results pertaining to $\Delta N = 2$ and $\Psi = F$ are presented in Table 5.

Table 4 ($n - q = 2$)

ΔN	N_s ($\Psi = f$)	N_s ($\Psi = F$)
1	5821	4569
2	8	15
3	12	13
4	7	10

Table 5 ($n - q = 2$, $\Delta N = 2$, $\Psi = F$)

N	N_0	x	y	z	f	Q
0	—	−2.0000	2.0000	2.0000	0.21 × 10^{1}	0.15 × 10^{1}
1	5	−0.3769	0.0132	1.3556	0.33 × 10^{0}	0.82 × 10^{0}
2	5	1.3559	0.9512	0.4357	0.43 × 10^{0}	0.21 × 10^{0}
3	3	1.7926	1.1752	0.6076	0.10 × 10^{0}	0.86 × 10^{0}
4	6	0.7711	0.9447	1.1140	0.30 × 10^{-1}	0.22 × 10^{0}
5	2	0.8335	0.8659	1.1078	0.35 × 10^{-2}	0.34 × 10^{-2}
6	4	1.0706	1.0316	0.9214	0.72 × 10^{-3}	0.25 × 10^{-3}
7	2	1.0538	1.0641	0.9234	0.39 × 10^{-4}	0.19 × 10^{-3}
8	3	1.0211	1.0225	0.9769	0.62 × 10^{-5}	0.17 × 10^{-4}
9	1	1.0218	1.0217	0.9770	0.39 × 10^{-6}	0.13 × 10^{-5}
10	2	1.0055	1.0056	0.9943	0.26 × 10^{-7}	0.85 × 10^{-7}
11	1	1.0055	1.0055	0.9943	0.15 × 10^{-8}	0.43 × 10^{-8}
12	2	1.0008	1.0008	0.9991	0.12 × 10^{-10}	0.24 × 10^{-10}
13	0	1.0008	1.0008	0.9991	0.97 × 10^{-11}	0.65 × 10^{-11}

Example 13.2. We consider the problem of minimizing the function

$$f = (x - y)^2 + (z - 1)^2 + (u - 1)^4 + (w - 1)^6 \tag{139}$$

subject to the constraints

$$xz^2 + \sin(u - w) - 1 = 0, \qquad y + x^4w^2 - 2 = 0 \tag{140}$$

Note that $n = 5$, $q = 2$, so that $n - q = 3$. This function admits the relative minimum $f = 0$ at the point defined by

$$x = 1, \quad y = 1, \quad z = 1, \quad u = 1, \quad w = 1 \tag{141}$$

The nominal point chosen for starting the algorithm is the point of coordinates

$$x = 1/\sqrt{2}, \quad y = 7/4, \quad z = \tfrac{1}{2}, \quad u = 2, \quad w = 2 \tag{142}$$

Table 6 ($n - q = 3$)

ΔN	N_* ($\Psi = f$)	N_* ($\Psi = F$)
1	50	112
2	10	10
3	8	11
4	10	12
5	12	12

Table 7 ($n - q = 3$, $\Delta N = 3$, $\Psi = F$)

N	N_0	x	y	z	u	w	f	Q
0	—	0.7071	1.7500	0.5000	2.0000	2.1600	0.33 × 10^{1}	0.58 × 10^{0}
1	5	0.9072	1.3220	1.0192	0.7926	0.4372	0.20 × 10^{0}	0.11 × 10^{0}
2	5	1.0700	1.0988	1.1376	0.7334	0.5725	0.30 × 10^{-1}	0.38 × 10^{-1}
3	3	1.0302	1.0681	1.0338	0.9032	0.8619	0.26 × 10^{-2}	0.44 × 10^{-2}
4	2	1.0387	1.0485	1.0105	0.9008	0.8728	0.18 × 10^{-3}	0.58 × 10^{-3}
5	3	1.0073	1.0074	1.0017	0.9868	0.9862	0.22 × 10^{-4}	0.11 × 10^{-3}
6	3	1.0063	1.0062	1.0038	0.9861	0.9869	0.14 × 10^{-5}	0.67 × 10^{-4}
7	1	1.0066	1.0069	1.0030	0.9885	0.9902	0.16 × 10^{-6}	0.73 × 10^{-5}
8	2	1.0030	1.0030	1.0000	0.9983	1.0026	0.51 × 10^{-7}	0.19 × 10^{-5}
9	1	1.0030	1.0030	1.0000	0.9983	1.0026	0.26 × 10^{-8}	0.76 × 10^{-6}
10	1	1.0030	1.0030	1.0000	0.9983	1.0028	0.22 × 10^{-9}	0.13 × 10^{-6}
11	1	1.0029	1.0029	1.0000	0.9985	1.0029	0.30 × 10^{-11}	0.61 × 10^{-10}

consistent with (140). Table 6 shows N_* for several values of ΔN and both $\Psi = f$ and $\Psi = F$. The detailed results pertaining to $\Delta N = 3$ and $\Psi = F$ are presented in Table 7.

Example 13.3. We consider the problem of minimizing the function

$$f = (x - y)^2 + (y - z)^4 + (z - u)^4 + (u - w)^6 \tag{143}$$

subject to the constraints

$$x + y^2 + z^2 - 3 = 0, \quad y - z^2 + u - 1 = 0, \quad zw - 1 = 0 \tag{144}$$

Note that $n = 5$, $q = 3$, so that $n - q = 2$. This function admits the relative minimum $f = 0$ at the point defined by

$$x = 1, \quad y = 1, \quad z = 1, \quad u = 1, \quad w = 1 \tag{145}$$

The nominal point chosen for starting the algorithm is the point of coordinates

$$x = 2, \quad y = \sqrt{2}, \quad z = -1, \quad u = 2 - \sqrt{2}, \quad w = \tfrac{1}{2} \tag{146}$$

consistent with (144). Table 8 shows N_* for several values of ΔN and both $\Psi = f$ and $\Psi = F$. The detailed results pertaining to $\Delta N = 2$ and $\Psi = F$ are presented in Table 9.

Table 8 ($n - q = 2$)

ΔN	N_s ($\Psi = f$)	N_s ($\Psi = F$)
1	29	11
2	10	11
3	11	11
4	10	12

Table 9 ($n - q = 2, \Delta N = 2, \Psi = F$)

N	N_s	x	y	z	v	w	f	Q
0	—	2.0000	1.4142	−1.0000	0.5557	0.5000	0.12×10^{2}	0.55×10^{2}
1	4	1.6940	1.6513	−0.0338	−0.0488	0.5279	0.20×10^{1}	0.46×10^{2}
2	4	1.6631	1.2390	0.3402	−0.0233	0.6012	0.10×10^{1}	0.37×10^{2}
3	6	1.0921	0.7044	1.0979	1.4489	0.9156	0.30×10^{0}	0.27×10^{2}
4	4	0.9059	0.8483	1.1118	1.2877	1.1037	0.83×10^{-1}	0.30×10^{0}
5	3	0.8563	0.9518	1.0719	1.1942	1.1677	0.23×10^{-1}	0.22×10^{-1}
6	3	1.0439	1.0147	0.9756	0.9376	0.9606	0.21×10^{-2}	0.26×10^{-2}
7	2	1.0017	1.0099	0.9927	0.9755	0.9982	0.36×10^{-3}	0.10×10^{-2}
8	2	1.0017	0.9999	0.9994	0.9968	0.9952	0.32×10^{-4}	0.89×10^{-3}
9	1	1.0005	1.0004	0.9995	0.9986	0.9994	0.80×10^{-5}	0.98×10^{-4}
10	1	0.9999	1.0000	0.9999	0.9999	1.0000	0.70×10^{-6}	0.29×10^{-4}
11	0	0.9999	0.9999	1.0000	1.0000	1.0000	0.20×10^{-10}	0.21×10^{-10}

14. Discussion and Conclusions

In the previous sections, a sequential algorithm is developed for minimizing a function $f(x)$ subject to the constraint $\varphi(x) = 0$. The algorithm is composed of the alternate succession of gradient phases and restoration phases. For the gradient phase, two versions are presented, one in which information at only points x is used (ordinary gradient version) and one in which information at both points x and $\bar{x}$ is used (conjugate gradient version). For the restoration phase, the criterion employed is that of the least-square change of the coordinates.

While the ordinary gradient version of the algorithm exhibits only asymptotic convergence, the conjugate gradient version exhibits quadratic convergence; this means that, for a quadratic function subject to a linear constraint, the minimum point is obtained in no more than $n - q$ iterations.

In order to illustrate the theory, several numerical examples are presented. The first three examples are concerned with a quadratic function subject to a linear constraint; the computer results confirm the quadratic convergence properties of the conjugate gradient version of the algorithm. The next three examples are concerned with general functions subject to general constraints; the computer results show the rapidly convergent properties of the conjugate gradient version of the algorithm and its superiority with respect to the ordinary gradient version.

In the theory as well as in the examples, the function $\Psi(x) = f(x)$ or the function $\Psi(x) = F(x, \lambda)$ is minimized along the search direction. However, the occurrence of cases where $\Psi(x)$ does not possess a relative minimum is conceivable. For these cases, an upper limit must be imposed on the stepsize α or the performance index $P(x)$.

In the numerical examples, the starting point was chosen so that $\varphi(x) = 0$. However, the present algorithms can be started even if $\varphi(x) \neq 0$. In this case, the first phase is a restoration phase rather than a gradient phase.

References

1. Miele, A., and Heideman, J. C., *Mathematical Programming for Constrained Minimal Problems, Part 1, Sequential Gradient-Restoration Algorithm*, Rice University, Aero-Astronautics Report No. 59, 1969.
2. Miele, A., Huang, H. Y., and Heideman, J. C., *Mathematical Programming for Constrained Minimal Problems, Part 2, Sequential Conjugate Gradient-Restoration Algorithm*, Rice University, Aero-Astronautics Report No. 61, 1969.
3. Miele, A., Huang, H. Y., and Cantrell, J. W., *Gradient Methods in Mathematical Programming, Part 1, Review of Previous Techniques*, Rice University, Aero-Astronautics Report No. 55, 1969.
4. Miele, A., Heideman, J. C., and Damoulakis, J. N., *The Restoration of Constraints in Holonomic and Nonholonomic Problems*, Journal of Optimization Theory and Applications, Vol. 3, No. 5, 1969.
5. Hestenes, M. R., and Stiefel, E., *Methods of Conjugate Gradients for Solving Linear Systems*, Journal of Research of the National Bureau of Standards, Vol. 49, No. 6, 1952.
6. Beckman, F. S., *The Solution of Linear Equations by the Conjugate Gradient Method*, Mathematical Methods for Digital Computers, Edited by A. Ralston and H. S. Wilf, John Wiley and Sons, New York, 1960.
7. Fletcher, R., and Reeves, C. M., *Function Minimization by Conjugate Gradients*, Computer Journal, Vol. 7, No. 2, 1964.

Additional Bibliography

Rosen, J. B., *The Gradient Projection Method for Nonlinear Programming, Part 2, Nonlinear Constraints*, SIAM Journal on Applied Mathematics, Vol. 9, No. 4, 1961.

Bryson, A. E., Jr., and Ho, Y.C., *Applied Optimal Control*, Chapter 1, Blaisdell Publishing Company, New York, 1969

PRINTED IN BELGIUM, BY THE ST. CATHERINE PRESS, LTD.

On the Method of Multipliers for Mathematical Programming Problems[1]

A. Miele,[2] P. E. Moseley,[3] A. V. Levy,[4] and G. M. Coggins[5]

Abstract. In this paper, the numerical solution of the basic problem of mathematical programming is considered. This is the problem of minimizing a function $f(x)$ subject to a constraint $\varphi(x) = 0$. Here, f is a scalar, x is an n-vector, and φ is a q-vector, with $q \leq n$.

The approach employed is based on the introduction of the augmented penalty function $\Pi(x,\lambda,k) = f(x) + \lambda^T\varphi(x) + k\varphi^T(x)\varphi(x)$. Here, the q-vector λ is an approximation to the Lagrange multiplier, and the scalar $k > 0$ is the penalty constant.

Previously, the augmented penalty function $\Pi(x,\lambda,k)$ was used by Hestenes[1] in his method of multipliers. In Hestenes' version, the method of multipliers involves cycles, in each of which the multiplier and the penalty constant are held constant. After the minimum of the augmented penalty function is achieved in any given cycle, the multiplier λ is updated, while the penalty constant k is held unchanged.

In this paper, two modifications of the method of multipliers are presented in order to improve its convergence characteristics. The improved convergence is achieved by (i) increasing the updating frequency so that the number of iterations in a cycle is shortened to $\Delta N = 1$ for the ordinary-gradient algorithm and the modified-quasilinearization algorithm and $\Delta N = n$ for the conjugate-gradient algorithm, (ii) imbedding Hestenes' updating rule for the multiplier λ

[1] Paper received February 18, 1971; in revised form, April 28, 1971. This research was supported by the National Science Foundation, Grant No. GP-32453. The authors are indebted to Messieurs E. E. Craig and A. Estrada for computational assistance.

[2] Professor of Astronautics and Mathematical Sciences, Rice University, Houston, Texas.

[3] Graduate Student in Aero-Astronautics, Rice University, Houston, Texas. Presently, Senior Research Engineer, Esso Production Research Company, Houston, Texas.

[4] Graduate Student in Aero-Astronautics, Rice University, Houston, Texas. Presently, Research Associate, University of Mexico, Mexico City, Mexico.

[5] Graduate Student in Aero-Astronautics, Rice University, Houston, Texas. Presently, Instructor, US Military Academy, West Point, New York.

into a one-parameter family and determining the scalar parameter β so that the error in the optimum condition is minimized, and (iii) updating the penalty constant k so as to cause some desirable effect in the ordinary-gradient algorithm, the conjugate-gradient algorithm, and the modified-quasilinearization algorithm. For the sake of identification, Hestenes' method of multipliers is called Method MM-1, the modification including (i) and (ii) is called Method MM-2, and the modification including (i), (ii), (iii) is called Method MM-3.

Evaluation of the theory is accomplished with seven numerical examples. The first example pertains to a quadratic function subject to linear constraints. The remaining examples pertain to non-quadratic functions subject to nonlinear constraints. Each example is solved with the ordinary-gradient algorithm, the conjugate-gradient algorithm, and the modified-quasilinearization algorithm, which are employed in conjunction with Methods MM-1, MM-2, and MM-3.

The numerical results show that (a) for given penalty constant k, Method MM-2 generally exhibits faster convergence than Method MM-1, (b) in both Methods MM-1 and MM-2, the number of iterations for convergence has a minimum with respect to k, and (c) the number of iterations for convergence of Method MM-3 is close to the minimum with respect to k of the number of iterations for convergence of Method MM-2. In this light, Method MM-3 has very desirable characteristics.

1. Introduction

Over the past several years, considerable work has been done on the numerical solution of the constrained minimization problem. This is the problem of minimizing a function $f(x)$ subject to a constraint $\varphi(x) = 0$. Here, f is a scalar, x is an n-vector, and φ is a q-vector, with $q < n$.

The methods employed are generally based on one of two basic ideas. One approach ensures constraint satisfaction, at least to first order, at the end of each iteration (see, for example, Refs. 1–3). The other approach depends on the construction of a sequence of special functions having, in the limit, an unconstrained minimum point coincident with the solution of the original constrained minimization problem. With regard to the latter approach, the standard penalty function method (see, for example, Refs. 4–6) and Hestenes' method of multipliers (Ref. 7) must be mentioned.

As the numerical experiments of Refs. 8–9 indicate, the method of multipliers generally exhibits faster convergence than the standard penalty function method. Crucial to the method of multipliers is the manner in which the multiplier λ is estimated and the penalty constant k is updated at the beginning of each cycle. These key questions are considered in this paper, whose objective is to enlarge the investigation of Ref. 10 and to develop techniques for improving the convergence characteristics of the method of multipliers. The resulting algorithm is called *modified method of multipliers*.

2. Statement of the Problem

We consider the problem of minimizing the function

$$f = f(x) \tag{1}$$

subject to the constraint

$$\varphi(x) = 0, \tag{2}$$

In the above equations, f is a scalar, x is an n-vector, and φ is a q-vector, with $q < n$. Here, all vectors are column vectors. It is assumed that the first and second partial derivatives of the functions f and φ exist and are continuous and that the constrained minimum exists.

2.1. First-Order Conditions. The previous problem can be recast as that of minimizing the augmented function

$$F(x, \lambda) = f(x) + \lambda^T \varphi(x) \tag{3}$$

subject to the constraint (2). The q-vector λ is the Lagrange multiplier, and the superscript T denotes the transpose of a matrix.

From theory of maxima and minima, it is known that the optimal solution must satisfy the relations

$$\varphi(x) = 0, \quad F_x(x, \lambda) = f_x(x) + \varphi_x(x)\lambda = 0, \tag{4}$$

which are a system of $n + q$ equations in x and λ. The subscript x denotes the gradient of a function; in this case, f_x and F_x are n-vectors, and φ_x is an $n \times q$ matrix, defined in such a way that its ith column is the gradient of the ith scalar component of φ with respect to x.

2.2. Approximate Solutions. In general, the system (4) is

nonlinear; consequently, approximate methods must be employed. Here, we introduce the scalar performance indexes

$$P(x) = \varphi^T(x)\varphi(x), \qquad Q(x,\lambda) = F_x^T(x,\lambda)F_x(x,\lambda), \tag{5}$$

which measure the errors in the constraint and the optimum condition, respectively. At the solution point, $P = 0$ and $Q = 0$, while $P > 0$ and/or $Q > 0$ for any approximation of the solution. When approximate methods are employed, they must ultimately lead to values of x, λ such that

$$P(x) \leq \epsilon_1, \qquad Q(x,\lambda) \leq \epsilon_2. \tag{6}$$

Alternatively, (6) can be replaced by

$$R(x,\lambda) \leq \epsilon_3, \tag{7}$$

where

$$R(x,\lambda) = P(x) + Q(x,\lambda) \tag{8}$$

denotes the cumulative error in the constraint and the optimum condition. Here, ϵ_1, ϵ_2, ϵ_3 are small, preselected numbers. Note that satisfaction of Ineq. (7) implies satisfaction of Ineqs. (6) if one chooses $\epsilon_1 = \epsilon_2 = \epsilon_3$.

3. Review of Penalty Function Methods

The penalty function method is based on the construction of a sequence of special functions having, in the limit, an unconstrained minimum point coincident with the solution of the original constrained minimization problem. In this section, two versions of the penalty function method are reviewed: (i) the standard penalty function method and (ii) the method of multipliers. Method (i) is based on the standard penalty function (9) and Method (ii) is based on the augmented penalty function (22).

3.1. Standard Penalty Function Method. This method is based on the consideration of the standard penalty function

$$U(x,k) = f(x) + k\varphi^T(x)\varphi(x). \tag{9}$$

This is obtained by adding to the function $f(x)$ a term quadratic in the constraint $\varphi(x)$, $k > 0$ being the penalty constant.

The problem of minimizing the function (1) subject to the constraint (2) is replaced by a sequence of unconstrained minimization problems. In each element of the sequence or cycle, one minimizes the function (9) with respect to x for given k. Therefore, theoretically speaking, the following necessary condition must be satisfied at the end of each cycle:

$$U_x(x,k) = f_x(x) + 2k\varphi_x(x)\varphi(x) = 0. \tag{10}$$

If the penalty constant k is arbitrary, the vector x which satisfies Eq. (10) is such that $\varphi(x) \neq 0$. However, if one defines the Lagrange multiplier to be

$$\lambda = 2k\varphi(x), \tag{11}$$

Eq. (10) reduces to

$$F_x(x,\lambda) = f_x(x) + \varphi_x(x)\lambda = 0, \tag{12}$$

meaning that the combination of x and λ thus obtained satisfies exactly the optimum condition.

In order to obtain constraint satisfaction, increasingly larger values of the penalty constant must be employed in successive cycles of the standard penalty function method. In this connection, let k_1 denote the penalty constant of the present cycle and k_2 denote the penalty constant of the next cycle, with $k_2 > k_1$. Because of the jump in k, the standard penalty function increases by the amount

$$U(x,k_2) - U(x,k_1) = (k_2 - k_1)P(x), \tag{13}$$

and the norm of the gradient of the standard penalty function takes the value[4]

$$U_x^T(x,k_2)\,U_x(x,k_2) = (k_2 - k_1)^2 P_x^T(x)\,P_x(x), \tag{14}$$

where

$$P(x) = \varphi^T(x)\varphi(x), \qquad P_x(x) = 2\varphi_x(x)\varphi(x). \tag{15}$$

The positiveness of the right-hand side of Eq. (13) is the key to the mechanism on which the standard penalty function method is based.

After a sufficient number of cycles, the constraint error can be made as small as desired providing the penalty constant has become sufficiently large. Theoretically speaking, the condition $\varphi(x) = 0$ is desired at convergence; consequently, the multiplier λ defined by Eq. (11) can be identical with the multiplier satisfying Eqs. (4), which is

[4] Note that $U_x(x,k_1) = 0$.

generally nonzero, only if $k \to \infty$. In a practical digital computer, this means that very large values of k are needed at convergence.

Numerical Implementation. From the above considerations, the following outline of the standard penalty function method emerges.

(a) The original constrained minimization problem is replaced by a sequence of unconstrained minimization problems.

(b) In each element of the sequence or cycle, the standard penalty function

$$U(x, k) = f(x) + k\varphi^T(x)\varphi(x) \tag{16}$$

is minimized with respect to x for given k. The minimum of $U(x, k)$ is achieved when the following stopping condition is satisfied:

$$U_x^T(x, k)\, U_x(x, k) \leq \epsilon_1, \tag{17}$$

where ϵ_1 is a small, preselected number.

(c) The solution point of any given cycle is chosen as the starting point of the next cycle of the standard penalty function method.

(d) For the next cycle, a higher value of the penalty constant is selected, one choice being

$$k_2 = ck_1, \tag{18}$$

where $c > 1$ is the penalty constant ratio.

(e) After updating the penalty constant, one returns to (b) and continues iteratively.

(f) The algorithm is terminated when the following stopping condition is satisfied:

$$\varphi^T(x)\varphi(x), \; U_x^T(x, k)\, U_x(x, k) \leq \epsilon_2, \tag{19}$$

where ϵ_2 is a small, preselected number.

Remark. At convergence of a cycle, the stopping condition (17) can be written as

$$Q(x, \lambda) \leq \epsilon_4, \tag{20}$$

where λ is given by Eq. (11). Analogously, at convergence of the algorithm, the stopping condition (19) becomes

$$H(x, \lambda) = P(x) + Q(x, \lambda) \leq \epsilon_5, \tag{21}$$

where λ is given by Eq. (11).

3.2. Method of Multipliers. This method is based on the consideration of the augmented penalty function

$$H(x, \lambda, k) = f(x) + \lambda^T\varphi(x) + k\varphi^T(x)\varphi(x). \tag{22}$$

This is obtained by adding to the penalty function $U(x, k)$ a term linear in the constraint $\varphi(x)$, the q-vector λ being an approximation to the Lagrange multiplier. The use of this function was suggested by Hestenes (Ref. 7) in order to circumvent the numerical difficulties associated with the extremely large values of the penalty constant required by the standard penalty function method.

The problem of minimizing the function (1) subject to the constraint (2) is replaced by a sequence of unconstrained minimization problems. In each element of the sequence or cycle, one minimizes the function (22) with respect to x for given λ and k. Therefore, theoretically speaking, the following necessary condition must be satisfied at the end of each cycle:

$$H_x(x, \lambda, k) = f_x(x) + \varphi_x(x)\lambda + 2k\varphi_x(x)\varphi(x) = 0 \tag{23}$$

and is equivalent to

$$H_x(x, \lambda, k) = -f_x(x) + \varphi_x(x)[\lambda + 2k\varphi(x)] = 0. \tag{24}$$

If the penalty constant k and the multiplier λ are arbitrary, the vector x which satisfies Eq. (24) is such that $\varphi(x) \neq 0$. However, by means of a proper updating rule, a new Lagrange multiplier can be found such that the optimum condition is satisfied exactly. In this connection, let λ_1 denote the Lagrange multiplier of the present cycle and λ_2 denote the Lagrange multiplier of the next cycle. If λ_2 is chosen to be

$$\lambda_2 = \lambda_1 + 2k\varphi(x), \tag{25}$$

Eq. (24) reduces to

$$F_x(x, \lambda_2) = f_x(x) + \varphi_x(x)\lambda_2 = 0. \tag{26}$$

meaning that the combination of x and λ_2 thus obtained satisfies exactly the optimum condition.

At the end of any given cycle, whenever the value of the Lagrange multiplier is changed from λ_1 to λ_2, the augmented penalty function increases by the amount

$$H(x, \lambda_2, k) - H(x, \lambda_1, k) = 2kP(x), \tag{27}$$

and the norm of the gradient of the augmented penalty function takes the value[3]

$$W_x^T(x, \lambda_2, k)\, W_x(x, \lambda_2, k) = k^2 P_x^T(x)\, P_x(x), \tag{28}$$

where $P(x)$ and $P_x(x)$ are given by Eqs. (15). The positiveness of the right-hand side of Eq. (27) is the key to the mechanism on which the method of multipliers is based.

The attention of the reader is called on the essential similarity between Eqs. (13) and (27). In the standard penalty function method, the drive toward constraint satisfaction is supplied by increasing the penalty constant from cycle to cycle. In the method of multipliers, the drive toward constraint satisfaction is supplied by changing the multiplier from cycle to cycle in accordance with Eq. (25).

While the standard penalty function method requires extremely large values of the penalty constant at convergence, this is not the case with the method of multipliers. In the latter method, convergence can be achieved even with moderate values of the penalty constant.

Numerical Implementation. From the above considerations, the following outline of the method of multipliers (Method MM-1) emerges.

(a) The original constrained minimization problem is replaced by a sequence of unconstrained minimization problems.

(b) In each element of the sequence or cycle, the augmented penalty function

$$W(x, \lambda, k) = f(x) + \lambda^T q(x) + k q^T(x)\, q(x) \tag{29}$$

is minimized with respect to x for given λ and k. The minimum of $W(x, \lambda, k)$ is achieved when the following stopping condition is satisfied:

$$W_x^T(x, \lambda, k)\, W_x(x, \lambda, k) \leq \epsilon_4, \tag{30}$$

where ϵ_4 is a small, preselected number.

(c) The solution point of any given cycle is chosen as the starting point of the next cycle of the method of multipliers.

(d) For the next cycle, the multiplier is updated according to the simple rule

$$\lambda_2 = \lambda_1 + 2kq(x). \tag{31}$$

[3] Note that $W_x(x, \lambda_2, k) = 0$.

(e) After updating the multiplier, one returns to (b) and continues iteratively.

(f) The algorithm is terminated when the following stopping condition is satisfied:

$$q^T(x)\, q(x) + W_x^T(x, \lambda, k)\, W_x(x, \lambda, k) \leq \epsilon_3, \tag{32}$$

where ϵ_3 is a small, preselected number.

(g) To start the algorithm some assumption concerning the multiplier is necessary. The simplest assumption is

$$\lambda = 0 \tag{33}$$

and is equivalent to stating that the augmented penalty function (29) and the standard penalty function (16) are identical for the first cycle of the algorithm.

Remark. At convergence of a cycle, the stopping condition (30) can be written as

$$Q(x, \lambda_2) \leq \epsilon_4, \tag{34}$$

where λ_2 is given by Eq. (31). Analogously, at convergence of the algorithm, the stopping condition (32) becomes

$$R(x, \lambda_2) = P(x) + Q(x, \lambda_2) \leq \epsilon_3, \tag{35}$$

where λ_2 is given by Eq. (31).

4. Modifications of the Method of Multipliers

The method of multipliers described in Section 3 has one drawback: a sequence of unconstrained minimization problems must be solved, each possibly requiring a large number of iterations ΔN. Consequently, the total number of iterations for convergence $N_c = \Sigma(\Delta N)$ may become excessive for practical applications.

In order to accelerate convergence, we explore here several modifications of the method of multipliers. These modifications are obtained by (i) shortening the length of a cycle, (ii) improving the estimate of the multiplier, and (iii) selecting the penalty constant in an appropriate fashion.

4.1. Updating Frequency. Let a cycle be defined as any sequence of iterations in which the multiplier λ and the penalty constant

k are held unchanged, while the vector x is viewed as unconstrained. Let ΔN denote the number of iterations in a cycle, regardless of whether complete convergence or incomplete convergence is achieved.

To shorten the cycle, we assign *a priori* the value of ΔN. Precisely, we choose ΔN as the *smallest* number of iterations compatible with the characteristics of the particular algorithm being considered. Therefore,

$$\Delta N = 1 \tag{36}$$

for the ordinary-gradient algorithm and the modified-quasilinearization algorithm and

$$\Delta N = n \tag{37}$$

for the conjugate-gradient algorithm. Therefore, the stopping condition (30) for a cycle is bypassed and is replaced by (36) for the ordinary-gradient algorithm and the modified-quasilinearization algorithm and by (37) for the conjugate-gradient algorithm.

4.2. Multiplier Estimate. Now, we assume that the cycle length ΔN is defined by Eq. (36) for the ordinary-gradient algorithm and the modified-quasilinearization algorithm and by Eq. (37) for the conjugate-gradient algorithm. Then, we inquire about ways in which the multiplier λ can be estimated.

First Estimate. We assume that Hestenes' updating rule (31) is employed at the end of any cycle of ΔN iterations. We note that the updated error in the optimum condition can be larger or smaller than the error prior to updating. Since an increase in the error $Q(x, \lambda)$ is not desirable, the multiplier λ might be chosen as follows:

$$\begin{aligned} \lambda_2 &= \lambda_1 \quad \text{if} \quad Q(x, \lambda_0) \geq Q(x, \lambda_1), \\ \lambda_2 &= \lambda_0 \quad \text{if} \quad Q(x, \lambda_0) < Q(x, \lambda_1), \end{aligned} \tag{38}$$

where λ_0 is given by

$$\lambda_0 = \lambda_1 + 2kP(x). \tag{39}$$

Second Estimate. The previous estimate can be improved if Hestenes' updating rule (31) is renounced and is replaced with the more general updating rule

$$\lambda_2 = \lambda_1 + 2\beta_1(x), \tag{40}$$

where β is a scalar parameter. This parameter must be determined so as to produce some optimum effect.

For given values of x and λ_1, a change in β causes a change in the updated multiplier λ_2. Consequently, the updated error in the optimum condition

$$Q(x, \lambda_2) = F_x^T(x, \lambda_2) F_x(x, \lambda_2) \tag{41}$$

changes. The optimum value of β is that which gives $Q(x, \lambda_2)$ the smallest value for given x and λ_1. After combining (40)-(41), we obtain the relations

$$\begin{aligned} Q(x, \lambda_1, \beta) &= [F_x(x, \lambda_1) + \beta P_x(x)]^T [F_x(x, \lambda_1) + \beta P_x(x)], \\ Q_\beta(x, \lambda_1, \beta) &= 2P_x^T(x) [F_x(x, \lambda_1) + \beta P_x(x)], \\ Q_{\beta\beta}(x, \lambda_1, \beta) &= 2P_x^T(x) P_x(x), \end{aligned} \tag{42}$$

the second of which vanishes for

$$\beta = -P_x^T(x) F_x(x, \lambda_1) / P_x^T(x) P_x(x). \tag{43}$$

This value of β minimizes $Q(x, \lambda_1, \beta)$, since $Q_{\beta\beta}(x, \lambda_1, \beta) > 0$ provided $P_x(x)$ does not vanish.

The method of multipliers with cycle stopping condition (30) replaced by (36) or (37) and with multiplier updating rule (31) replaced by (40) and (43) is called modified method of multipliers or Method MM-2. For this method, the following comments are pertinent.

(i) Equations (40) and (43) imply that

$$P_x^T(x) F_x(x, \lambda_2) = 0. \tag{44}$$

Therefore, the optimum value of the parameter β is such that the gradient of the constraint error and the gradient of the updated augmented function are orthogonal.

(ii) The relation between the present updating rule and Hestenes' updating rule can be obtained as follows. Let the gradient of the augmented penalty function prior to updating be rewritten as

$$W_x(x, \lambda_1, k) = F_x(x, \lambda_1) + kP_x(x). \tag{45}$$

Then, combining (43) and (45) yields the relation

$$\beta = k - P_x^T(x) W_x(x, \lambda_1, k) / P_x^T(x) P_x(x), \tag{46}$$

which shows that

$$\beta \leq k, \tag{47}$$

providing

$$W_2(x, \lambda_1, k) \geq 0 \quad \text{or} \quad P_x^T(x)\, W_2(x, \lambda_1, k) \geq 0. \tag{48}$$

Clearly, the present updating rule and Hestenes'-updating rule are identical if applied at complete convergence, that is, at a point where Ineq. (30) is satisfied.

(iii) Equations (40) and (43) are to be employed at the beginning of each cycle of ΔN iterations, including the first cycle. However, for the first cycle, λ_1 is not defined. This being the case, an assumption is necessary, and the simplest assumption is

$$\lambda_1 = 0. \tag{49}$$

(iv) Whenever (40) and (43) are employed, the algorithm should be started at a point where $q(x) \neq 0$.

Third Estimate. In the previous section, the Lagrange multiplier was estimated within the frame of the one-parameter family (40). An even better estimate can be obtained by removing this limitation and minimizing the performance index $Q(x, \lambda)$ with respect to the q-vector λ for given x. After observing that

$$\begin{aligned} Q(x, \lambda) &= [f_x(x) + \varphi_x(x)\lambda]^T [f_x(x) + \varphi_x(x)\lambda], \\ Q_1(x, \lambda) &= 2\varphi_x^T(x)[f_x(x) + \varphi_x(x)\lambda], \\ Q_2(x, \lambda) &= 2\varphi_x^T(x)\varphi_x(x), \end{aligned} \tag{50}$$

we see that the optimal multiplier is defined by the relation

$$\varphi_x^T(x)\varphi_x(x)\lambda + \varphi_x^T(x) f_x(x) = 0, \tag{51}$$

which is a system of q scalar equations in the q components of the multiplier λ. The solution of (51) minimizes Q, since the matrix (50-3) is positive definite. The following comments are pertinent.

(i) On premultiplication by $\varphi_x^T(x)$, Eq. (51) leads to

$$P_x^T(x)\, P_x(x, \lambda) = 0. \tag{52}$$

Therefore, the optimum value of the multiplier λ is such that the gradient of the constraint error and the gradient of the penalized cost

multiplier λ compatible with any given position vector x (Ref. 3). However, its use requires the solution of a system of q linear equations in q unknowns. This being the case, the decision on whether to use (40) and (43) or (51) should be made on the basis of the particular algorithm employed.

(iii) For the ordinary-gradient algorithm and the conjugate-gradient algorithm, the displacement vector Δx is computed without solving a system of linear equations. Hence, the estimation of λ should be made with (40) and (43) rather than (51).

(iv) For the modified-quasilinearization algorithm, the displacement Δx is computed by solving a system of n linear equations in n unknowns. Hence, it is optional to estimate λ with (40) and (43) or (51). In the sequel, the estimate given by (40) and (43) is employed.

4.3. Penalty Constant Estimate. Now, the question arises as to whether the penalty constant can be selected in such a manner as to improve the convergence characteristics of Method MM-2. In this section, two techniques are presented for updating the penalty constant at the end of a cycle, one suitable for the ordinary-gradient algorithm and one suitable for the conjugate-gradient algorithm and the modified-quasilinearization algorithm. Method MM-2 with Eqs. (40) and (43) completed by a relation updating the penalty constant is called Method MM-3.

Ordinary-Gradient Algorithm. When this algorithm is employed, the multiplier-updating rule (40) and (43) induces an interesting characteristic: a descent property in the constraint error $P(x)$. This characteristic is utilized in this section to establish an updating rule for the penalty constant.

While the ordinary-gradient algorithm is described in detail in Section 5, we note here that the displacement Δx leading from the nominal point x to the varied point $\bar{x}$ produces a change in the constraint error $P(x)$. To first order, this change is given by

$$\delta P(x) = -\alpha k P_x^T(x)\, P_x(x). \tag{53}$$

where α is the stepsize, and k is the penalty constant. Therefore, $\delta P(x) \leq 0$, since $\alpha \geq 0$ and $k \geq 0$. This result guarantees that

$$P(\bar{x}) \leq P(x) \tag{54}$$

stant, we select k in such a way that, on the average, the constraints are satisfied to first order. Thereby, we determine k from the relation

$$\delta P(x) = -2\alpha P(x). \tag{55}$$

Comparing (53) and (55), we see that the appropriate value of the penalty constant should be

$$k = 2P(x)/P_x^T(x)\,P_x(x). \tag{56}$$

Since both the numerator and the denominator of the right-hand side of Eq. (56) contain equal powers of $\varphi(x)$, the penalty constant k varies slowly along the algorithm and is finite at convergence. For the particular case of a single scalar constraint ($q = 1$), Eq. (56) reduces to

$$k = 1/2\varphi_x^T(x)\,\varphi_x(x), \tag{57}$$

where $\varphi_x(x)$ is now an n-vector.

Conjugate-Gradient Algorithm and Modified-Quasilinearization Algorithm. The penalty constant estimate developed for the ordinary-gradient algorithm is based on the descent property (53) and the descent requirement (55). It produces a slowly varying penalty constant (56), which is finite at convergence.

For the conjugate-gradient algorithm and the modified-quasilinearization algorithm, the penalty constant (56) is not desirable for the reasons indicated below. Consider a quadratic function $f(x)$ and a linear constraint $\varphi(x)$. Regardless of the value of k, the augmented penalty function $H(x, \lambda, k)$ is quadratic in x. Theoretically speaking, the optimality condition (24) is satisfied exactly after n iterations of the conjugate-gradient algorithm and after one iteration of the modified quasilinearization algorithm; hence, the theoretical optimum value of the penalty constant should be $k = \infty$, so that it guarantees simultaneous satisfaction of the constraint equation $\varphi(x) = 0$ at the end of a cycle. In a practical digital computer, this result means that large values of the penalty constant should be employed if fast convergence is desired.

If the function $f(x)$ is nonquadratic and/or the constraint $\varphi(x)$ is nonlinear, the above reasoning is approximately true near the solution of the constrained minimization problem. This leads to the concept of programming k so as to achieve moderate values far away from the solution and large values near the solution, even though these large values are not as large as those needed with the standard penalty function method.

Possible choices of the penalty constant satisfying the above property are the following:

$$k_0 = |\lambda_2^T \varphi(x)|/P(x) \tag{58}$$

or

$$k_0 = \sqrt{[(\varphi_x(x)\,\lambda_2]^T\,[\varphi_x(x)\,\lambda_2]/P_x^T(x)\,P_x(x)}. \tag{59}$$

If Eq. (58) is employed, the order of magnitude of the linear term and the quadratic term appearing in the augmented penalty function is the same. If Eq. (59) is employed, the order of magnitude of the gradient of the linear term and the gradient of the quadratic term appearing in the augmented penalty function is the same.*

If the convergence requirements on the constraint error $P(x)$ and the error in the optimum condition $Q(x, \lambda)$ are the same, it might be desirable to reduce $P(x)$ and $Q(x, \lambda)$ at approximately the same rate. With this idea in mind, we propose the following updating rule for the penalty constant:

$$\begin{aligned} &k_2 = \min(k_0, k_1) \quad \text{if} \quad P(x) \leq Q(x, \lambda_2), \\ &k_2 = \max(k_0, \pi k_1) \quad \text{if} \quad P(x) > Q(x, \lambda_2), \end{aligned} \tag{60}$$

where $\pi \geq 1$, k_0 is given by (58) or (59), k_1 is the penalty constant prior to updating, and k_2 is the penalty constant after updating. Obviously, (60-1) prevents an increase of the penalty constant if the constraint error is relatively small. Conversely, (60-2) prevents a decrease of the penalty constant if the constraint error is relatively large.

Equation (60) is to be employed at the beginning of each cycle of ΔN iterations, including the first cycle. However, for the first cycle, k_1 is not defined. This being the case, an assumption is necessary, and the simplest assumption is

$$k_1 = k_0. \tag{61}$$

4.4. Summary of Methods. In this section, we summarize the combination of methods arising from the previous discussion, as follows.

(i) Method MM-1 is characterized by updating decision (30), multiplier estimate (31), and penalty constant unchanged throughout a particular algorithm.

(ii) Method MM-2 is characterized by updating decision (36) or

* Whenever (58) or (59) is employed, the algorithm should be started at a point where $\varphi(x) \neq 0$.

(37), multiplier estimate (40) and (43), and penalty constant unchanged throughout a particular algorithm.

(iii) Method MM-3 is characterized by updating decision (36) or (37), multiplier estimate (40) and (43), and penalty constant estimate (56) or (58) and (60).

Remark. The updating decision (36) is to be employed with the ordinary-gradient algorithm and the modified-quasilinearization algorithm, and the updating decision (37) is to be employed with the conjugate-gradient algorithm. Also, the penalty constant estimate (56) is to be employed with the ordinary-gradient algorithm, and the penalty constant estimate (58) and (60) is to be employed with the conjugate-gradient algorithm and the modified-quasilinearization algorithm.

5. Unconstrained Minimization Algorithms

In this section, the unconstrained minimization algorithms employed to compute the displacement vector Δx in connection with Methods MM-2 and MM-3 are described. They are the ordinary-gradient algorithm, the conjugate-gradient algorithm, and the modified-quasilinearization algorithm. All of these algorithms make use of the augmented penalty function

$$H(x, \lambda, k) = F(x, \lambda) + kP(x), \tag{62}$$

where

$$F(x, \lambda) = f(x) + \lambda^T \psi(x), \qquad P(x) = \psi^T(x)\,\psi(x), \tag{63}$$

and are employed with this understanding: in each cycle of ΔN iterations, the multiplier λ and the penalty constant k are held unchanged, and the vector x is viewed as unconstrained.

5.1. Ordinary-Gradient Algorithm. Let x denote the nominal point, $\bar{x}$ the varied point, Δx the displacement leading from the nominal point to the varied point, and α the stepsize. With this understanding, the ordinary-gradient algorithm is represented by

$$F_x(x, \lambda) = f_x(x) + \psi_x(x)\,\lambda, \tag{64-1}$$

$$P_x(x) = 2\psi_x(x)\,\psi(x), \tag{64-2}$$

$$H'_x(x, \lambda, k) = F_x(x, \lambda) + kP_x(x), \tag{64-3}$$

$$p = H'_x(x, \lambda, k), \tag{64-4}$$

$$\Delta x = -\alpha p, \tag{64-5}$$

$$\bar{x} = x + \Delta x. \tag{64-6}$$

For given nominal point x, multiplier λ, and penalty constant k, Eqs. (64) constitute a complete iteration leading to the varied point $\bar{x}$, providing one specifies the stepsize α.

Descent Properties. To first order, the changes in the functions $H(x, \lambda, k)$ and $P(x)$ are given by

$$\delta H(x, \lambda, k) = H'^T_x(x, \lambda, k)\,\Delta x, \qquad \delta P(x) = P_x^T(x)\,\Delta x, \tag{65}$$

which, in the light of (64), become

$$\delta H(x, \lambda, k) = -\alpha H'^T_x(x, \lambda, k)\,H'_x(x, \lambda, k),$$
$$\delta P(x) = -\alpha P_x^T(x)[F_x(x, \lambda) + kP_x(x)]. \tag{66}$$

Recalling Eq. (44), we see that the following orthogonality condition holds:

$$P_x^T(x)\,F_x(x, \lambda) = 0, \tag{67}$$

so that

$$\delta H(x, \lambda, k) = -\alpha H'^T_x(x, \lambda, k)\,H'_x(x, \lambda, k),$$
$$\delta P(x) = -\alpha k P_x^T(x)\,P_x(x). \tag{68}$$

Since the right-hand sides of (68) are negative, the following inequalities can be enforced for α sufficiently small:

$$H(\bar{x}, \lambda, k) < H(x, \lambda, k), \qquad P(\bar{x}) < P(x). \tag{69}$$

While enforcement of (69-1) is mandatory, enforcement of (69-2) is optional, but can be used in order to give greater stability to the ordinary-gradient algorithm.

5.2. Conjugate-Gradient Algorithm. Let x denote the nominal point, $\hat{x}$ the previous point, $\bar{x}$ the varied point, Δx the displacement leading from the nominal point to the varied point, p the present search direction, $\hat{p}$ the previous search direction, γ the directional coefficient, and α the stepsize. With this understanding, the conjugate-gradient algorithm is represented by

$$F_x(x, \lambda) = f_x(x) + \psi_x(x)\,\lambda, \tag{70-1}$$

$$P_x(x) = 2\psi_x(x)\,\psi(x), \tag{70-2}$$

$$W'_x(x, \lambda, k) = F'_x(x, \lambda) + kP'_x(x), \tag{70-3}$$

$$\gamma = W'_x{}^T(x, \lambda, k)\, W'_x(x, \lambda, k) / W'_x{}^T(\bar{x}, \lambda, k)\, W'_x(\bar{x}, \lambda, k), \tag{70-4}$$

$$p = W'_x(x, \lambda, k) + \gamma \hat{p}, \tag{70-5}$$

$$\Delta x = -\alpha p, \tag{70-6}$$

$$\bar{x} = x + \Delta x. \tag{70-7}$$

For given nominal point x, multiplier λ, directional coefficient γ, and penalty constant k, Eqs. (70) constitute a complete iteration leading to the varied point $\bar{x}$, providing one specifies the stepsize α. For the first iteration of a cycle, Eq. (70-4) is bypassed and is replaced by $\gamma = 0$.

Descent Properties. To first order, the changes in the function $W(x, \lambda, k)$ and $P(x)$ are given by Eqs. (65) which, in the light of (70), become

$$\begin{aligned} \delta W(x, \lambda, k) &= -\alpha W'_x{}^T(x, \lambda, k)\,[W'_x(x, \lambda, k) + \gamma \hat{p}], \\ \delta P(x) &= -\alpha P'_x{}^T(x)\,[F'_x(x, \lambda) + kP'_x(x) + \gamma \hat{p}]. \end{aligned} \tag{71}$$

For the first iteration of a cycle, relation (67) must be applied in conjunction with $\gamma = 0$, leading to

$$\begin{aligned} \delta W(x, \lambda, k) &= -\alpha W'_x{}^T(x, \lambda, k)\, W'_x(x, \lambda, k), \\ \delta P(x) &= -\alpha k P'_x{}^T(x)\, P'_x(x). \end{aligned} \tag{72}$$

For subsequent iterations, relation (67) does not hold and $\gamma \neq 0$. However, since the previous stepsize is optimized, the following orthogonality relation can be invoked:

$$W'_x{}^T(x, \lambda, k)\, \hat{p} = 0, \tag{73}$$

with the consequence that

$$\begin{aligned} \delta W(x, \lambda, k) &= -\alpha W'_x{}^T(x, \lambda, k)\, W'_x(x, \lambda, k), \\ \delta P(x) &= -\alpha P'_x{}^T(x)\,[F'_x(x, \lambda) + kP'_x(x) + \gamma \hat{p}]. \end{aligned} \tag{74}$$

Inspection of (72) and (74) shows that the descent property on the augmented penalty function

$$W(\bar{x}, \lambda, k) < W(x, \lambda, k) \tag{75}$$

can be enforced for all iterations of a cycle regardless of the value of k.

On the other hand, the descent property on the constraint error

$$P(\bar{x}) < P(x) \tag{76}$$

can be enforced for the first iteration of a cycle regardless of the value of k and for subsequent iterations only if k is sufficiently large.

5.3. Modified-Quasilinearization Algorithm. Let x denote the nominal point, $\bar{x}$ the varied point, Δx the displacement leading from the nominal point to the varied point, p the search direction, $\rho = \pm 1$ the direction factor, and α the stepsize. With this understanding, the modified-quasilinearization algorithm is represented by

$$F_x(x, \lambda) = f_x(x) + \varphi_x(x)\, \lambda, \tag{77-1}$$

$$P_x(x) = 2\varphi_x(x)\, \varphi(x), \tag{77-2}$$

$$W'_x(x, \lambda, k) = F_x(x, \lambda) + kP_x(x), \tag{77-3}$$

$$F_{xx}(x, \lambda) = f_{xx}(x) + \varphi_{xx}(x)\, \lambda, \tag{77-4}$$

$$P_{xx}(x) = 2[\varphi_{xx}(x)\, \varphi(x) + \varphi_x(x)\, \varphi'_x{}^T(x)], \tag{77-5}$$

$$W'_{xx}(x, \lambda, k) = F_{xx}(x, \lambda) + kP_{xx}(x), \tag{77-6}$$

$$W'_{xx}(x, \lambda, k)\, A + W'_x(x, \lambda, k) = 0, \tag{77-7}$$

$$\rho = \operatorname{sign}[W'_x{}^T(x, \lambda, k)\, A], \tag{77-8}$$

$$p = \rho A, \tag{77-9}$$

$$\Delta x = -\alpha p, \tag{77-10}$$

$$\bar{x} = x + \Delta x. \tag{77-11}$$

For given nominal point x, multiplier λ, and penalty constant k, Eqs. (77) constitute a complete iteration leading to the varied point $\bar{x}$, providing one specifies the stepsize α.

Descent Properties. To first order, the changes in the functions $W(x, \lambda, k)$ and $P(x)$ are given by Eqs. (65) which, in the light of (77), become

$$\begin{aligned} \delta W(x, \lambda, k) &= -\alpha \operatorname{sign}[W'_x{}^T(x, \lambda, k)\, A]\, W'_x{}^T(x, \lambda, k)\, A, \\ \delta P(x) &= -\alpha \operatorname{sign}[F'_x{}^T(x, \lambda)\, A + kP'_x{}^T(x)\, A]\, P'_x{}^T(x)\, A. \end{aligned} \tag{78}$$

Inspection of (78) shows that the descent property on the augmented penalty function

$$W(\bar{x}, \lambda, k) < W(x, \lambda, k) \tag{79}$$

can be enforced regardless of the value of k. On the other hand, the descent property on the constraint error

$$P(\bar{x}) < P(x) \tag{80}$$

can be enforced for any k if

$$P_x'(x, \lambda)\, d[P_x'(x)]\, d > 0 \tag{81}$$

and only for k sufficiently large if

$$P_x'(x, \lambda)\, d[P_x'(x)]\, d < 0. \tag{82}$$

6. Stepsize Determination

For all of the previous algorithms, the position vector at the end of any step can be written as

$$\bar{x} = x - \alpha p, \tag{83}$$

where p denotes the search direction. This is a one-parameter family of varied points $\bar{x}$, for which the augmented penalty function (62) takes the form

$$\Pi(\bar{x}, \lambda, k) = \Pi(x - \alpha p, \lambda, k) = \Pi(\alpha). \tag{84}$$

A precise search to be employed with the conjugate-gradient algorithm and an approximate search to be employed with the ordinary-gradient algorithm and the modified-quasilinearization algorithm are described below.

Precise Search. We now assume that a minimum of $\Pi(\alpha)$ exists. Then, we employ some one-dimensional search scheme (for instance, quadratic interpolation, cubic interpolation, or quasilinearization) to determine the value of α for which

$$\Pi_\alpha'(\alpha) = 0. \tag{85}$$

Ideally, this procedure should be used iteratively until the modulus of the slope satisfies any of the following inequalities:

$$|\Pi_\alpha'(\alpha)| \leqslant \epsilon_a \quad \text{or} \quad |\Pi_\alpha'(\alpha)| \leqslant \epsilon_b |\Pi_\alpha'(0)|, \tag{86}$$

where ϵ_a and ϵ_b are small, preselected numbers. Of course, the value of α satisfying Ineq. (86) must be such that

$$\Pi(\alpha) < \Pi(0). \tag{87}$$

Approximate Search. Since the rigorous determination of α might require excessive computing time, one might recommend solving Eq. (85) with a particular degree of precision and determine the stepsize in a noniterative fashion. For instance, one might employ a bisection procedure on α, starting from a reference value $\alpha = \alpha_R$, until satisfaction of Ineq. (87) occurs. For the ordinary gradient algorithm, the reference stepsize α_R can be chosen to be the first optimum value of α supplied by the search procedure. For the modified-quasilinearization algorithm, the reference stepsize α_R can be chosen to be $\alpha_R = 1$.

Remark. Optionally, Ineq. (87) can be completed by the additional inequality

$$P(\alpha) < P(0), \tag{88}$$

which is designed to give greater stability to the algorithm. Inequality (88) can be enforced in the ordinary-gradient algorithm for any k. In the conjugate-gradient algorithm and the modified-quasilinearization algorithm, Ineq. (88) can be enforced only for k sufficiently large.

7. Experimental Conditions

In order to evaluate the theory, seven numerical examples were explored. Each example was solved with the ordinary-gradient algorithm, the conjugate-gradient algorithm, and the modified-quasilinearization algorithm, which were employed in conjunction with Methods MM-1, MM-2, and MM-3. All of the algorithms were programmed in FORTRAN IV, and the numerical results were obtained using a Burroughs B-5500 computer and double-precision arithmetic.

Starting Point of the Algorithm. For all of the examples, the nominal point chosen to start the algorithm was defined by

$$x_1 = x_2 = \cdots = x_n = 2, \tag{89}$$

where n denotes the dimension of the vector x.

Convergence of the Algorithm. Convergence of an algorithm was defined through the inequality

$$P(x) + Q(x, \lambda) \leqslant 10^{-4} \tag{90}$$

for the ordinary-gradient algorithm and the inequality

$$P(x) + Q(x, \lambda) \leqslant 10^{-12} \tag{91}$$

for the conjugate-gradient algorithm and the modified-quasilinearization algorithm.

Nonconvergence of the Algorithm. Conversely, nonconvergence of an algorithm was defined by means of the inequalities

$$\text{(a)} \quad \begin{cases} N > 1000 & \text{for the ordinary-gradient algorithm,} \\ N > 200 & \text{for the conjugate-gradient algorithm,} \\ N > 100 & \text{for the modified-quasilinearization algorithm,} \end{cases} \tag{92}$$

or

$$\text{(b)} \quad N_s > 20 \tag{93}$$

or

$$\text{(c)} \quad M > 0.4 \times 10^{38}. \tag{94}$$

Here, N is the iteration number, N_s is the number of bisections of the stepsize α required to satisfy Ineq. (87) [and optionally Ineq. (88)], and M is the modulus of any of the quantities employed in the algorithm. Satisfaction of Ineq. (92) indicates divergence or extreme slowness of convergence; satisfaction of Ineq. (93) indicates extreme smallness of the displacement Δx; and satisfaction of Ineq. (94) indicates exponential overflow. Each of these situations is undesirable.

Convergence of a Cycle. When Method MM-1 was employed, convergence of a cycle was defined through the inequality

$$W_x^T(x, \lambda_1, k)\, W_x(x, \lambda_1, k) \leqslant 10^{-8} \tag{95}$$

for the ordinary-gradient algorithm and the inequality

$$W_x^T(x, \lambda_1, k)\, W_x(x, \lambda_1, k) \leqslant 10^{-12} \tag{96}$$

for the conjugate-gradient algorithm and the modified-quasilinearization algorithm.

When Methods MM-2 and MM-3 were employed, convergence of a cycle was defined by

$$\Delta N = 1 \tag{97}$$

for the ordinary-gradient algorithm and the modified-quasilinearization algorithm and by either

$$\Delta N = n \tag{98}$$

or

$$W_x^T(x, \lambda_1, k)\, W_x(x, \lambda_1, k) \leqslant 10^{-12}, \tag{99}$$

whichever occurred first, for the conjugate-gradient algorithm.

Search Technique: For the ordinary-gradient algorithm, an approximate search was employed. This consisted of one-step, corrected quasilinearization, followed by a bisection process until the inequality

$$W(\alpha) < W(0) \tag{100}$$

was satisfied.

For the conjugate-gradient algorithm, a precise search was employed. This consisted of multistep, corrected quasilinearization such that, in any given step, the inequality

$$W(\alpha) < W(\alpha_0) \tag{101}$$

was satisfied, where α_0 is the nominal stepsize and α is the varied stepsize. The search was started with $\alpha_0 = 0$ and was terminated when the following stopping condition was satisfied:

$$W_x^T(\alpha) \leqslant W_x^T(0) \times 10^{-4}. \tag{102}$$

For the modified-quasilinearization algorithm, an approximate search was employed. This consisted of assigning the value

$$\alpha = 1 \tag{103}$$

to the stepsize, followed by a bisection process until Ineq. (100) was satisfied.

Penalty Constant Estimate. For Method MM-3, the penalty constant k was estimated with Eq. (56) for the ordinary-gradient algorithm and with Eqs. (58) and (60), with $n = 1$, for the conjugate-gradient algorithm and the modified-quasilinearization algorithm.

8. Numerical Examples

In this section seven numerical examples are described. The first example pertains to a quadratic function subject to linear constraints. The remaining examples pertain to nonquadratic functions subject to nonlinear constraints.

Example 8.1. Consider the problem of minimizing the function

$$f = (x_1 - x_4)^4 + (x_2 + x_3 - 2)^4 + (x_4 - 1)^4 + (x_5 - 1)^4 \tag{104}$$

subject to the constraints

$$x_1 + 3x_2 = 0, \qquad x_3 + x_4 - 2x_5 = 0, \qquad x_2 - x_5 = 0. \tag{105}$$

This function admits the relative minimum $f = 4.0930$ at the point defined by

$$x_1 = -0.7674, \quad x_2 = 0.2558, \quad x_3 = 0.6279, \quad x_4 = -0.1162, \quad x_5 = 0.2558 \tag{106}$$

and

$$\lambda_1 = 2.0165, \quad \lambda_2 = 2.2325, \quad \lambda_3 = -5.9534. \tag{107}$$

Example 8.2. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^4 \tag{108}$$

subject to the constraint

$$x_1(1 + x_2^2) + x_3^4 - 4 - 3\sqrt{2} = 0. \tag{109}$$

This function admits the relative minimum $f = 0.3256 \times 10^{-1}$ at the point defined by

$$x_1 = 1.1018, \quad x_2 = 1.1966, \quad x_3 = 1.5352 \tag{110}$$

and

$$\lambda_1 = -0.1072 \times 10^{-1}. \tag{111}$$

Example 8.3. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_3 - 1)^4 + (x_4 - 1)^4 + (x_5 - 1)^6 \tag{112}$$

subject to the constraints

$$x_1^2 x_4 + \sin(x_4 - x_5) - 2\sqrt{2} = 0, \quad x_2 + x_3^4 x_4^2 - 8 - \sqrt{2} = 0. \tag{113}$$

This function admits the relative minimum $f = 0.2415$ at the point defined by

$$x_1 = 1.1661, \quad x_2 = 1.1821, \quad x_3 = 1.3802, \quad x_4 = 1.5968, \quad x_5 = 0.6109 \tag{114}$$

and

$$\lambda_1 = -0.8553 \times 10^{-1}, \quad \lambda_2 = -0.3187 \times 10^{-1}. \tag{115}$$

Example 8.4. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^3 + (x_3 - x_4)^4 + (x_4 - x_5)^4 \tag{116}$$

subject to the constraints

$$x_1^2 + x_2^3 + x_3^2 - 2 + 3\sqrt{2} = 0, \quad x_1 - x_2^2 + x_3^4 + 2 - 2\sqrt{2} = 0, \quad x_1 x_5 - 2 = 0. \tag{117}$$

This function admits the relative minimum $f = 0.7877 \times 10^{-1}$ at the point defined by

$$x_1 = 1.1911, \quad x_2 = 1.3626, \quad x_3 = 1.4728, \quad x_4 = 1.6350, \quad x_5 = 1.6790 \tag{118}$$

and

$$\lambda_1 = -0.3882 \times 10^{-1}, \quad \lambda_2 = -0.1672 \times 10^{-1}, \quad \lambda_3 = -0.2879 \times 10^{-8}. \tag{119}$$

Example 8.5. Consider the problem of minimizing the function

$$f = 0.01(x_1 - 1)^2 + (x_2 - x_1^2)^2 \tag{120}$$

subject to the inequality constraint

$$x_1 \leqslant -1. \tag{121}$$

Introduce the auxiliary variable x_3 defined by

$$x_1 + x_3^2 + 1 = 0. \tag{122}$$

Then, the previous problem can be recast as that of minimizing the function (120) subject to the equality constraint (122). The function (120) admits the relative minimum $f = 0.04$ at the point defined by

$$x_1 = -1, \quad x_2 = 1, \quad x_3 = 0 \tag{123}$$

and

$$\lambda_1 = 0.04. \tag{124}$$

Example 8.6. Consider the problem of minimizing the function

$$f = -x_3 \tag{125}$$

subject to the inequality constraints

$$x_2 \geqslant x_1^2, \quad x_3 \leqslant x_1^2. \tag{126}$$

Introduce the auxiliary variables x_4 and x_5 defined by

$$x_2 - x_1^2 - x_4^2 = 0, \quad x_1^2 - x_3 - x_5^2 = 0. \tag{127}$$

Then, the previous problem can be recast as that of minimizing the function (125) subject to the equality constraints (127). The function (125) admits the relative minimum $f = -1$ at the point defined by

$$x_1 = 1, \quad x_2 = 1, \quad x_3 = 0, \quad x_4 = 0 \tag{128}$$

and

$$\lambda_1 = -1, \quad \lambda_2 = -1. \tag{129}$$

Example 8.7. Consider the problem of minimizing the function

$$f = \log x_3 - x_4 \tag{130}$$

subject to the equality constraint

$$x_3^2 + x_2^2 - 4 = 0 \tag{131}$$

and the inequality constraint

$$x_3 \geq 1. \tag{132}$$

Introduce the auxiliary variable x_1 defined by

$$x_3 = 1 + x_1^2. \tag{133}$$

Then, the previous problem can be recast as that of minimizing the function

$$f = \log(1 + x_1^2) - x_4 \tag{134}$$

subject to the equality constraint

$$(1 + x_1^2)^2 + x_2^2 - 4 = 0. \tag{135}$$

Note that x_3 has been eliminated from the problem and can be computed *a posteriori* with (133). The function (134) admits the relative minimum $f = -\sqrt{3}$ at the point defined by

$$x_1 = 0, \quad x_2 = \sqrt{3}, \quad x_3 = 1 \tag{136}$$

and

$$\lambda_1 = 1/2\sqrt{3}. \tag{137}$$

9. Results and Conclusions

The examples described in Section 8 were solved with Hestenes' method of multipliers (Method MM-1) and the modified method of multipliers (Methods MM-2 and MM-3) according to the experimental

Table 1. Method MM-1, ordinary-gradient algorithm, number of iterations N_ϵ.

	Examples						
ϵ	8.1	8.2	8.3	8.4	8.5	8.6	8.7
10^{-3}	(a)	24	565	580	(a)	(a)	44
10^{-4}	664	73	847	193	516	(a)	17
10^{-5}	350	646	(a)	431	58	946	35
10^{-6}	863	(a)	(a)	377	761	762	155
10^{-8}	(a)	(a)	(a)	(a)	92	746	774

(a) Number of iterations exceeded 1000.

Table 2. Method MM-2, ordinary-gradient algorithm, number of iterations N_ϵ.

	Examples						
ϵ	8.1	8.2	8.3	8.4	8.5	8.6	8.7
10^{-3}	(a)	14	135	102	138	455	54
10^{-4}	394	21	433	94	39	318	11
10^{-5}	94	661	(a)	306	33	278	32
10^{-6}	266	998	(a)	(a)	161	129	49
10^{-8}	(a)	(a)	(a)	(a)	194	(a)	133

(a) Number of iterations exceeded 1000.

Table 3. Method MM-3, ordinary-gradient algorithm, number of iterations N_ϵ.

	Examples						
ϵ	8.1	8.2	8.3	8.4	8.5	8.6	8.7
Eq. (56)	88	13	193	101	20	275	17

Table 4. Method MM-1, conjugate-gradient algorithm, number of iterations N_*.

	Examples						
k	8.1	8.2	8.3	8.4	8.5	8.6	8.7
10^{-3}	(a)	25	(a)	(a)	93	(a)	58
10^{-1}	(a)	28	142	138	(a)	(a)	16
10^{0}	(a)	37	103	84	40	184	14
10^{1}	49	54	145	81	71	52	11
10^{3}	20	76	176	95	164	33	18

(a) Number of iterations exceeded 200.

Table 5. Method MM-2, conjugate-gradient algorithm, number of iterations N_*.

	Examples						
k	8.1	8.2	8.3	8.4	8.5	8.6	8.7
10^{-3}	(a)	16	(a)	(a)	(a)	(a)	55
10^{-1}	(a)	16	76	93	63	(a)	13
10^{0}	(a)	33	79	42	49	150	10
10^{1}	47	48	110	50	61	36	12
10^{3}	20	71	172	49	(a)	28	16

(a) Number of iterations exceeded 200.

Table 6. Method MM-3, conjugate-gradient algorithm, number of iterations N_*.

	Examples						
k	8.1	8.2	8.3	8.4	8.5	8.6	8.7
Eqs. (58), (60)	43	15	75	53	32	50	14

Table 7. Method MM-1, modified-quasilinearization algorithm, number of iterations N_*.

	Examples						
k	8.1	8.2	8.3	8.4	8.5	8.6	8.7
10^{-3}	(a)	14	(a)	(a)	56	(a)	66
10^{-1}	(a)	11	31	34	21	(a)	25
10^{0}	46	14	37	17	14	86	20
10^{1}	10	13	19	13	26	29	23
10^{3}	4	13	25	22	46	20	35

(a) Number of iterations exceeded 100.

Table 8. Method MM-2, modified-quasilinearization algorithm, number of iterations N_*.

	Examples						
k	8.1	8.2	8.3	8.4	8.5	8.6	8.7
10^{-3}	(a)	9	90	(a)	(c)	(a)	46
10^{-1}	(a)	8	20	26	(c)	(a)	16
10^{0}	46	11	(d)	12	16	58	14
10^{1}	10	10	17	11	21	19	20
10^{3}	4	12	25	21	45	16	33

(a) Number of iterations exceeded 100.
(c) Exponential overflow.
(d) Algorithm converged to a different relative minimum.

Table 9. Method MM-3, modified-quasilinearization algorithm, number of iterations N_*.

	Examples						
k	8.1	8.2	8.3	8.4	8.5	8.6	8.7
Eqs. (38), (60)	9	9	13	9	13	21	12

conditions outlined. These methods were employed in conjunction with the ordinary-gradient algorithm, the conjugate-gradient algorithm, and the modified-quasilinearization algorithm. For Methods MM–1 and MM–2, several values of the penalty constant, ranging between 10^{+3} and 10^{8}, were considered.

The numerical results are presented in Tables 1–9, where the number of iterations required for convergence, N_c, is shown. Tables 1–3 refer to the ordinary-gradient algorithm, Tables 4–6 refer to the conjugate-gradient algorithm, and Tables 7–9 refer to the modified-quasilinearization algorithm.

Comparison of Methods MM–1 and MM–2 shows that, for given k, Method MM–2 generally exhibits faster convergence than Method MM–1. For both Methods MM–1 and MM–2, the number of iterations for convergence has a minimum with respect to k.

Concerning Method MM–3, the number of iterations for convergence is close to the minimum with respect to k of the number of iterations for convergence of Method MM–2. In this light, Method MM–3 has very desirable characteristics.

Remark 9.1. In Section 4.2, several ways to estimate the Lagrange multiplier λ were presented. The estimate represented by Eqs. (40) and (43) was preferred to that represented by Eq. (51) in order to avoid solving a set of q linear equations in q unknowns. The above statement is significant if one employs the ordinary-gradient algorithm and the conjugate-gradient algorithm, since the computation of the displacement vector Δx is made bypassing the solution of sets of linear equations. On the other hand, if one employs the modified-quasilinearization algorithm, a case can be made for using the estimate (51) for the multiplier, since the computation of the displacement vector Δx requires the solution of a set of n linear equations in n unknowns.

For the sake of discussion, let the modified-quasilinearization algorithm be employed and let the following terminology be used: (a) Method MM–3 is the modified method of multipliers with Lagrange multiplier represented by (40) and (43) and penalty constant represented by (58) and (60); and (b) Method MM–4 is the modified method of multipliers with Lagrange multiplier represented by (51) and penalty constant represented by (58) and (60).

As a point of interest, the seven numerical examples of Section 8 were solved employing both Methods MM–3 and MM–4. The comparative results are given in Table 10, where the number of iterations at convergence, N_c, is shown. As expected, for a single scalar constraint ($q = 1$), Methods MM–3 and MM–4 converge to the solution in the same number of iterations. This is precisely the case with Examples 8.2, 8.5, and 8.7. On the other hand, if several constraints are present, Method MM–4 generally converges to the solution in a smaller number of iterations than Method MM–3.

Table 10. Modified-quasilinearization algorithm, number of iterations N_c.

Method	Examples						
	8.1	8.2	8.3	8.4	8.5	8.6	8.7
MM-3	9	9	13	9	13	21	12
MM-4	8	9	10	11	13	15	12

Remark 9.2. Both Methods MM–3 and MM–4 employed in conjunction with the modified-quasilinearization algorithm are interesting in that they generally lead to a minimum point (or at most a saddle point), while standard quasilinearization may also lead to a maximum point.[8] To verify this concept, the following example was considered: Extremize the function

$$f = \sin(\pi x_1/12)\cos(\pi x_2/16) \tag{138}$$

subject to the constraint

$$4x_1 - 3x_2 = 0. \tag{139}$$

This function has a relative minimum $f = -0.5$ at

$$x_1 = -3, \quad x_2 = -4, \quad \text{and} \quad \lambda_1 = -\pi/96. \tag{140}$$

and a relative maximum $f = 0.5$ at

$$x_1 = 3, \quad x_2 = 4, \quad \text{and} \quad \lambda_1 = -\pi/96. \tag{141}$$

The following nominal point was considered:

$$x_1 = 2, \quad x_2 = 2. \tag{142}$$

It was found that Methods MM–3 and MM–4 employed in conjunction with the modified-quasilinearization algorithm led to the relative

[8] Standard quasilinearization involves the solution of a system of $n + q$ linear equation in $n + q$ unknowns (see, for example, Ref. 3).

minimum (140) in $N_* = 5$ iterations. On the other hand, standard quasilinearization (Ref. 3) let to the relative maximum (141) in $N_* = 3$ iterations.

Remark 9.3. In Ref. 11, it was suggested that Hestenes' method of multipliers could be accelerated by employing Eq. (31) at the beginning of each iteration, rather than at the beginning of each cycle. With particular reference to the conjugate-gradient algorithm, this technique has the disadvantage of producing discontinuities in the augmented penalty function $\Pi(\mathbf{r}, \lambda, k)$ within a cycle of $\Delta N = n$ iterations. To verify this point, numerical experiments were carried out for the seven examples of Section 8 and one value of the penalty constant, namely, $k = 1$. It was found that, in the majority of the examples, the algorithm of Ref. 11 had not converged to the solution within the required accuracy (91) in the imposed limit of 200 iterations.

References

1. Miele, A., Huang, H. Y., and Heideman, J. C., *Sequential Gradient-Restoration Algorithm for the Minimization of Constrained Functions, Ordinary and Conjugate Gradient Versions*, Journal of Optimization Theory and Applications, Vol. 4, No. 4, 1969.
2. Miele, A., Heideman, J. C., and Levy, A. V., *Combined Conjugate Gradient-Restoration Algorithm for Mathematical Programming Problems*, Ricerche di Automatica, Vol. 2, No. 2, 1971.
3. Miele, A., and Levy, A. V., *Modified Quasilinearization and Optimal Initial Choice of the Multipliers, Part 1, Mathematical Programming Problems*, Journal of Optimization Theory and Applications, Vol. 6, No. 5, 1970.
4. Kelley, H. J., *Method of Gradients*, Optimization Techniques, Edited by G. Leitmann, Academic Press, New York, 1962.
5. Bryson, A. E., Jr., and Ho, Y. C., *Applied Optimal Control*, Blaisdell Publishing Company, Waltham, Massachusetts, 1969.
6. Fiacco, A. V., and McCormick, G. P., *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley and Sons, New York, 1968.
7. Hestenes, M. R., *Multiplier and Gradient Methods*, Journal of Optimization Theory and Applications, Vol. 4, No. 5, 1969.
8. Miele, A., Moseley, P. E., and Cragg, E. E., *Numerical Experiments on Hestenes' Method of Multipliers for Mathematical Programming Problems*, Rice University, Aero-Astronautics Report No. 85, 1971.

Penalty Constant Used in the Penalty Function Method for Mathematical Programming Problems, Rice University, Aero-Astronautics Report No. 90, 1972.

10. Miele, A., Moseley, P. E., and Cragg, E. E., *A Modification of the Method of Multipliers for Mathematical Programming Problems*, Techniques of Optimization, Edited by A. V. Balakrishnan, Academic Press, New York, 1972.
11. Tripathi, S. S., and Narendra, K. S., *Constrained Optimization Using Multiplier Methods*, Journal of Optimization Theory and Applications, Vol. 9, No. 1, 1972.

COMPARISON OF SEVERAL GRADIENT ALGORITHMS FOR MATHEMATICAL PROGRAMMING PROBLEMS

by

Angelo MIELE, J. L. TIETZE and A. V. LEVY
Department of Mechanical and Aerospace Engineering and Material Science
Rice University
Houston, Texas

COMPARISON OF SEVERAL GRADIENT ALGORITHMS FOR MATHEMATICAL PROGRAMMING PROBLEMS

by

ANGELO MIELE, J.L. TIETZE and A.V. LEVY

This work is dedicated by the senior author to Professor Carlo Ferrari with deep friendship and profound admiration for his many achievements in aerospace engineering and applied mathematics throughout the years.

Abstract

In this paper, the numerical solution of the basic problem of mathematical programming is considered. This is the problem of minimizing a function $f(x)$ subject to a constraint $\varphi(x) = 0$. Here, f is a scalar, x an n-vector, and φ a q-vector, with $q \leq n$.

Six variations of the sequential gradient-restoration algorithm and the combined gradient-restoration algorithm are considered, and their relative efficiency (in terms of number of iterations for convergence) is evaluated. The variations being considered are as follows:

(i) SGRA-CR, sequential gradient-restoration algorithm, complete restoration,
(ii) SGRA-IR, sequential gradient-restoration algorithm, incomplete restoration,
(iii) SGRA-OR, sequential gradient-restoration algorithm, optional restoration,
(iv) CGRA-NR, combined gradient-restoration algorithm, no restoration,
(v) CGRA-AR, combined gradient-restoration algorithm, alternate restoration,
(vi) CGRA-OR, combined gradient-restoration algorithm, optional restoration.

Evaluation of these algorithms is accomplished through eight numerical examples. The first two examples pertain to quadratic functions subject to linear constraints. The remaining examples pertain to nonquadratic functions subject to nonlinear constraints. The results indicate that (a) the inclusion of a restoration phase is necessary for

rapid convergence and (b) the algorithms with alternate restoration or optional restoration are the most efficient among those considered here.

1. Introduction

In previous papers (refs.1,2,3), two basic algorithms for the minimization of constrained functions were developed: the sequential gradient-restoration algorithm (SGRA) and the combined gradient-restoration algorithm (CGRA). The former is an iterative algorithm which consists of the alternate succession of gradient phases and restoration phases; the latter is an iterative algorithm in which the gradient phase and the restoration phase are combined in a single phase.

In the gradient phase of SGRA, one generates a displacement δx lowering the value of the function, while avoiding excessive constraint violation; in the restoration phase of SGRA, one generates a displacement δx restoring the constraint to a predetermined accuracy, while avoiding excessive change in the value of the function. On the other hand, in the gradient-restoration phase of CGRA, one generates a displacement δx lowering the value of the augmented function, while simultaneously reducing the constraint violation.

In this paper, six variations of the sequential gradient-restoration algorithm and the combined gradient-restoration algorithm are considered, and their relative efficiency (in terms of number of iterations for convergence) is evaluated through eight numerical examples. The variations being considered are indicated below:

(i) SGRA-CR, sequential gradient-restoration algorithm, complete restoration,
(ii) SGRA-IR, sequential gradient-restoration algorithm, incomplete restoration,
(iii) SGRA-OR, sequential gradient-restoration algorithm, optional restoration,
(iv) CGRA-NR, combined gradient-restoration algorithm, no restoration,
(v) CGRA-AR, combined gradient-restoration algorithm, alternate restoration,
(vi) CGRA-OR, combined gradient-restoration algorithm, optional restoration.

2. Statement of the Problem

We consider the problem of minimizing the function

$$f = f(x) , \tag{1}$$

subject to the constraint

$$\varphi(x) = 0 , \tag{2}$$

where f is a scalar, x an n-vector, and φ a q-vector, with $q < n$. Here, all vectors are column vectors. It is assumed that the first and second partial derivatives of the functions $f(x)$ and $\varphi(x)$ exist and are continuous and that the constrained minimum exists.

2.1. First-Order Conditions. From theory of maxima and minima, it is known that the above problem is equivalent to that of minimizing the augmented function

$$F(x,\lambda) = f(x) + \lambda^T \varphi(x) , \tag{3}$$

subject to the constraint (2). Here, the q-vector λ is the Lagrange multiplier and the superscript T denotes the transpose of a matrix. If

$$F_x(x,\lambda) = f_x(x) + \varphi_x(x)\lambda \tag{4}$$

denotes the gradient of the augmented function, the optimum solution for x and λ must satisfy the relations

$$\varphi(x) = 0 , \qquad F_x(x,\lambda) = 0 , \tag{5}$$

which are a system of $n + q$ equations in the $n + q$ components of x and λ. In Eqs. (4)-(5), the gradients f_x and F_x denotes n-vectors and the matrix φ_x is $n \times q$.

2.2. Approximate Solutions. Since the system (5) is generally nonlinear, approximate methods must be employed. In this connection, we introduce here the scalar performance indexes

$$P(x) = \varphi^T(x)\,\varphi(x) , \qquad Q(x,\lambda) = F_x^T(x,\lambda)\,F_x(x,\lambda) , \tag{6}$$

which measure the errors in the constraint and the optimum condition, respectively. Then, we observe that $P = 0$ and $Q = 0$ for the optimum solution, while $P>0$ and/or $Q>0$ for any approximation to the solution. When approximate methods are used, they must ultimately lead to

values of x and λ such that

$$P(x) \leq \epsilon_1 \quad , \quad Q(x, \lambda) \leq \epsilon_2 \quad . \tag{7}$$

Alternatively, (7) can be replaced by

$$R(x, \lambda) \leq \epsilon_3 \quad , \tag{8}$$

where

$$R(x, \lambda) = P(x) + Q(x, \lambda) \tag{9}$$

denotes the cumulative error in the constraint and the optimum condition. In (7)-(8), ϵ_1, ϵ_2, ϵ_3 are small, preselected numbers. Note that, if one chooses $\epsilon_1 = \epsilon_2 = \epsilon_3$, satisfaction of Ineq. (8) implies satisfaction of Ineqs. (7).

3. Description of the Algorithms

In this section, the algorithms being investigated are described.

SGRA-CR. *Sequential gradient-restoration algorithm, complete restoration.* This algorithm consists of the alternate succession of gradient phases and restoration phases.

The gradient phase is started providing

$$P(x) \leq \epsilon_1 \quad . \tag{10}$$

It involves a single iteration, in which the augmented function is reduced subject to an upper limit for the constraint error, that is,

$$F(\bar{x}, \lambda) < F(x, \lambda) \quad , \quad P(\bar{x}) \leq \epsilon_4 \quad . \tag{11}$$

The symbol x denotes the nominal point, $\bar{x}$ the varied point, and λ the Lagrange multiplier.

The restoration phase is started providing

$$P(x) > \epsilon_1 \quad . \tag{12}$$

It involves several iterations, in each of which the constraint error is reduced, that is,

$$P(\bar{x}) < P(x) \quad . \tag{13}$$

The restoration phase is terminated whenever Ineq. (10) is satisfied.

Remark. The algorithm is started with a gradient phase if Ineq. (10) is satisfied or a restoration phase if Ineq. (10) is violated. Normally, a gradient phase is followed by a restoration phase. Occasionally, the gradient phase is followed by another gradient phase, that is, the restoration phase is bypassed; this is precisely the case whenever Ineq. (10) is satisfied.

SGRA-IR. *Sequential gradient-restoration algorithm, incomplete restoration.* This algorithm consists of the alternate succession of gradient phases and restoration phases.

The gradient phase is started regardless of whether Ineq. (10) is satisfied. It involves a single iteration, in which the augmented function is reduced subject to an upper limit on the constraint error, that is,

$$F(\bar{x}, \lambda) < F(x, \lambda) \quad , \quad P(\bar{x}) \leq P(x) + \epsilon_4 \quad . \tag{14}$$

The restoration phase is started only if Ineq. (12) is satisfied. It involves a single iteration, in which the constraint error is reduced, in accordance with Ineq. (13).

The starting condition and the bypassing condition for SGRA-IR are identical with those of SGRA-CR (see Remark).

SGRA-OR. *Sequential gradient-restoration algorithm, optional restoration.* This algorithm consists of the alternate succession of gradient phases and restoration phases.

The gradient phase is started providing

$$Z(x, \lambda) \leq 1 \quad , \tag{15}$$

where the parameter Z is defined by

$$Z = \epsilon P(x)/Q(x, \lambda) \quad , \tag{16}$$

with

$$\epsilon = \epsilon_2/\epsilon_1 \quad . \tag{17}$$

It involves a single iteration, in which the augmented function is reduced in accordance with Ineqs. (14).

The restoration phase is started providing

$$Z(x, \lambda) > 1 \quad . \tag{18}$$

It involves several iterations, in each of which the constraint error is reduced in accordance with Ineq.(13). The restoration phase is terminated whenever Ineq.(15) is satisfied.

The bypassing condition for SGRA-OR is identical with that of SGRA-CR (see Remark).

CGRA-NR. *Combined gradient-restoration algorithm, no restoration.* In this algorithm, the gradient phase and the restoration phase are combined together in a single phase. It involves a single iteration, in which the augmented function is reduced in accordance with Ineqs.(14).

CGRA-AR. *Combined gradient-restoration algorithm, alternate restoration.* This algorithm consists of the alternate succession of combined gradient-restoration phases and restoration phases.

The combined gradient-restoration phase is started regardless of whether Ineq.(10) is satisfied. It involves a single iteration, in which the augmented function is reduced in accordance with Ineqs.(14).

The restoration phase is started only if Ineq.(12) is satisfied. It involves a single iteration, in which the constraint error is reduced in accordance with Ineq.(13).

The starting condition and the bypassing condition for CGRA-AR are identical with those of SGRA-CR (see Remark).

CGRA-OR. *Combined gradient-restoration algorithm, optional restoration.* This algorithm consists of the alternate succession of combined gradient-restoration phases and restoration phases.

The combined gradient-restoration phase is started providing Ineq.(15) is satisfied. It involves a single iteration, in which the augmented function is reduced in accordance with Ineqs.(14).

The restoration phase is started providing Ineq.(18) is satisfied. It involves several iterations, in each of which the constraint error is reduced in accordance with Ineq.(13). The restoration phase is terminated whenever Ineq.(15) is satisfied.

The bypassing condition for CGRA-OR is identical with that of SGRA-CR (see Remark).

Remark. For the algorithms with optional restoration, the multiplier λ appearing in (15)-(18) is computed as follows. For SGRA-OR, Eq.(19-1) must be solved with $C_1 = 1$ and $C_2 = 0$. For CGRA-OR, Eq.(19-1) must be solved with $C_1 = 1$ and $C_2 = 1$.

4. Generalized Algorithm

Let x denote the nominal point, $\bar{x}$ the varied point, Δx the displacement leading from the nominal point to the varied point, and α the stepsize. With this understanding, the previous algorithms can be represented in the following generalized form:

$$\varphi_x^T(x)\varphi_x(x)\lambda + C_1 \varphi_x^T(x) f_x(x) - C_2 \varphi(x) = 0 , \tag{19-1}$$

$$p = C_1 f_x(x) + \varphi_x(x)\lambda , \tag{19-2}$$

$$\Delta x = -\alpha p , \tag{19-3}$$

$$\bar{x} = x + \Delta x . \tag{19-4}$$

For given nominal point x and constants C_1 and C_2, Eqs.(19) constitute a complete iteration leading to the varied point $\bar{x}$, providing one specifies the stepsize α. The constants C_1 and C_2 depend on the particular algorithm and take the values given in Table 1.
The detailed derivation of Eqs.(19) is presented in refs.1,2,3 and, hence, is not repeated here.

Table 1. Characteristic constants.

Algorithm	Phase	C_1	C_2
SGRA	Gradient	1	0
	Restoration	0	1
CGRA	Gradient-restoration	1	1
	Restoration	0	1

5. Stepsize Determination

For all of the previous algorithms, the position vector at the end of any step can be written as

$$\bar{x} = x - \alpha p , \tag{20}$$

where p denotes the search direction, which is given by (19-2). This is a one-parameter family of varied points $\bar{x}$, for which the augmented function (3), the constraint error (6-1), and the error in the optimum condition (6-2) take the form

$$F(\bar{x}, \lambda) = F(x - \alpha p, \lambda) = \bar{F}(\alpha) \quad , \tag{21}$$

$$P(\bar{x}) = P(x - \alpha p) = \bar{P}(\alpha) \quad , \tag{22}$$

$$Q(\bar{x}, \lambda) = Q(x - \alpha p, \lambda) = \bar{Q}(\alpha) \quad . \tag{23}$$

For the gradient phase of a SGRA-algorithm or the combined gradient-restoration phase of a CGRA-algorithm, Inequs. (11) and (14) can be written in the general form

$$\bar{F}(\alpha) < \bar{F}(0) \quad , \quad \bar{P}(\alpha) \leq \bar{P}(0) + \epsilon_1 \quad . \tag{24}$$

Their satisfaction can be ensured by employing a bisection process, starting from a suitably chosen reference stepsize

$$\alpha = \alpha_0 \quad . \tag{25}$$

For the determination of the reference stepsize, see Section 6.

For the restoration phase of a SGRA-algorithm or a CGRA-algorithm, Ineq. (1)) can be written as

$$\bar{P}(\alpha) < \bar{P}(0) \quad . \tag{26}$$

Its satisfaction can be ensured by employing a bisection process, starting from the reference stepsize

$$\alpha = 1 \quad . \tag{27}$$

This value reduces the constraint error $P(x)$ to zero, if the constraint function $g(x)$ is linear in x.

6. Reference Stepsize

The search technique outlined in Section 5 for the gradient stepsize employs a bisection process, starting from the reference stepsize (25), until satisfaction of Inequ. (24) occurs. A procedure useful to determine this reference stepsize is outlined here and is based on a quadratic

representation of the augmented function associated with the one-parameter family of solutions (20).

Let the function $\bar{F}(\alpha)$ be represented in the quadratic form

$$\bar{F}(\alpha) = k_0 + k_1\alpha + k_2\alpha^2 \quad , \tag{28}$$

and let the coefficients of the quadratic be determined so as to match the values of the ordinate and the slope at $\alpha = 0$ and the value of the ordinate at $\alpha = 1$. This yields the relations

$$\bar{F}(0) = k_0 \quad , \quad \bar{F}_\alpha(0) = k_1 \quad , \quad \bar{F}(1) = k_0 + k_1 + k_2 \quad , \tag{29}$$

which imply that

$$k_0 = \bar{F}(0) \quad , \quad k_1 = -\bar{Q}(0) \quad , \quad k_2 = \bar{F}(1) - \bar{F}(0) + \bar{Q}(0) \quad . \tag{30}$$

With the coefficients known, the following possibilities arise:

$$\text{(i) } k_2 > 0 \quad \text{or} \quad \text{(ii) } k_2 \leq 0 \quad . \tag{31}$$

In Case (i), the quadratic function (28) has a minimum for the following value of the gradient stepsize:

$$\alpha = -k_1/2k_2 \quad . \tag{32}$$

In Case (ii), the quadratic function (28) decreases monotonically with α. This suggests the use of the following reference values for the gradient stepsize:

$$\alpha_0 = -k_1/2k_2 \quad \text{if} \quad k_2 > 0 \quad ,$$
$$\alpha_0 = 1 \quad \text{if} \quad k_2 \leq 0 \quad . \tag{33}$$

7. Experimental Conditions

In order to evaluate the previous algorithms, eight numerical examples were considered. The first two examples pertain to quadratic functions subject to linear constraints. The remaining examples pertain to nonquadratic functions subject to nonlinear constraints. Each example was solved with the three versions of SGRA and the three versions of CGRA

outlined in Section 3. All of the algorithms were programmed in FORTRAN IV, and the numerical results were obtained using a Burroughs B - 5500 computer and double-precision arithmetic.

Starting Point. For all of the examples, the nominal point chosen to start an algorithm was defined by

$$x_1 = x_2 = \ldots = x_n = 2, \tag{34}$$

where n denotes the dimension of the vector x.

Search Technique. The determination of the gradient stepsize and the restoration stepsize was performed in accordance with Sections 5 and 6. For the gradient phase, the stepsize α was subject to the inequalities

$$\tilde{P}(\alpha) < \tilde{P}(0), \quad \tilde{P}(\alpha) \leq \tilde{P}(0) + 1 . \tag{35}$$

For the restoration phase, the stepsize was subject to the inequality

$$\tilde{P}(\alpha) < \tilde{P}(0) . \tag{36}$$

Convergence. Convergence of an algorithm was defined through the inequalities

$$P(x) \leq 10^{-8}, \quad Q(x,\lambda) \leq 10^{-4}. \tag{37}$$

Nonconvergence. Conversely, nonconvergence of an algorithm was defined by means of the inequalities

$$N > 100 , \tag{38-1}$$

(a)

or

$$N_s > 20 , \tag{38-2}$$

(b)

or

$$M > 0.4 \times 10^{69} . \tag{38-3}$$

(c)

Here, N is the iteration number, N_s is the number of bisections of the stepsize α required to satisfy Ineq. (35) or (36), and M is the modulus of any of the quantities employed in the algorithm. Satisfaction of Ineq. (38-1) indicates divergence or extreme slowness of convergence; satisfaction of Ineq. (38-2) indicates extreme smallness of the displacement dx; and satisfaction of Ineq. (38-3) indicates exponential overflow. Each of these situations is undesirable.

8. Numerical Examples

In this section, eight numerical examples are described. The first two examples pertain to quadratic functions subject to linear constraints. The remaining examples pertain to nonquadratic functions subject to nonlinear constraints.

Example 8.1. Consider the problem of minimizing the function

$$f = (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2 , \tag{39}$$

subject to the constraints

$$x_1 + 3x_2 = 0 , \quad x_3 + x_4 - 2x_5 = 0, \quad x_2 - x_5 = 0 . \tag{40}$$

This function admits the relative minimum $f = 4.0930$ at the point defined by

$$x_1 = -0.7674, \quad x_2 = 0.2558, \quad x_3 = 0.6279, \quad x_4 = -0.1162, \quad x_5 = 0.2558 \tag{41}$$

and

$$\lambda_1 = 2.0465 , \quad \lambda_2 = 2.2325 , \quad \lambda_3 = -5.9534 . \tag{42}$$

Example 8.2. Consider the problem of minimizing the function

$$f = (4x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2 , \tag{43}$$

subject to the constraints

$$x_1 + 3x_2 = 0 , \quad x_3 + x_4 - 2x_5 = 0, \quad x_2 - x_5 = 0 . \tag{44}$$

This function admits the relative minimum $f = 5.3266$ at the point defined by

$$x_1 = -0.9455 \times 10^{-1}, \quad x_2 = 0.3151 \times 10^{-1}, \quad x_3 = 0.5157 , \tag{45}$$

$$x_4 = -0.4527, \quad x_5 = 0.3151 \times 10^{-1}$$

and

$$\lambda_1 = 3.2779, \quad \lambda_2 = 2.9054 , \quad \lambda_3 = -7.7478 . \tag{46}$$

Example 8.3. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^4 , \tag{47}$$

subject to the constraint

$$x_1(1 + x_2^2) + x_3^4 - 4 - 3\sqrt{2} = 0 . \tag{48}$$

This function admits the relative minimum $f = 0.3256 \times 10^{-1}$ at the point defined by

$$x_1 = 1.1048, \quad x_2 = 1.1966, \quad x_3 = 1.5352 , \tag{49}$$

and

$$\lambda_1 = -0.1072 \times 10^{-1} . \tag{50}$$

Example 8.4. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_3 - 1)^2 + (x_4 - 1)^4 + (x_5 - 1)^6 , \tag{51}$$

subject to the constraints

$$x_1^2 x_4 + \sin(x_4 - x_5) - 2\sqrt{2} = 0, \quad x_2 + x_3^4 x_4^2 - 8 - \sqrt{2} = 0. \tag{52}$$

This function admits the relative minimum $f = 0.2415$ at the point defined by

$$x_1 = 1.1661, \quad x_2 = 1.1821, \quad x_3 = 1.3802, \quad x_4 = 1.5060, \quad x_5 = 0.6109 \tag{53}$$

and

$$\lambda_1 = -0.8553 \times 10^{-1} , \quad \lambda_2 = -0.3187 \times 10^{-1} . \tag{54}$$

Example 8.5. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + (x_3 - x_4)^4 + (x_4 - x_5)^4 , \tag{55}$$

subject to the constraints

$$x_1 + x_2^2 + x_3^4 - 2 - 3\sqrt{2} = 0, \quad x_2 - x_3^2 + x_4 + 2 - 2\sqrt{2} = 0, \quad x_1 x_5 - 2 = 0 . \tag{56}$$

This function admits the relative minimum $f = 0.7877 \times 10^{-1}$ at the point defined by

$$x_1 = 1.1911, \quad x_2 = 1.3626, \quad x_3 = 1.4728, \quad x_4 = 1.6150, \quad x_5 = 1.6790 \tag{57}$$

and

$$\lambda_1 = -0.3882 \times 10^{-1} , \quad \lambda_2 = -0.1672 \times 10^{-1}, \quad \lambda_3 = -0.2879 \times 10^{-1}. \tag{58}$$

Example 8.6. Consider the problem of minimizing the function

$$f = 0.01(x_1 - 1)^2 + (x_2 - x_3^2)^2 , \tag{59}$$

subject to the inequality constraint

$$x_1 \leq -1 . \tag{60}$$

Introduce the auxiliary variable x_3 defined by

$$x_1 + x_3^2 + 1 = 0 . \tag{61}$$

Then, the previous problem can be recast as that of minimizing the function (59) subject to the equality constraint (61). The function (59) admits the relative minimum $f = 0.04$ at the point defined by

$$x_1 = -1, \quad x_2 = 1 , \quad x_3 = 0 \tag{62}$$

and

$$\lambda_1 = 0.04 . \tag{63}$$

Example 8.7. Consider the problem of minimizing the function

$$f = -x_1 , \tag{64}$$

subject to the inequality constraints

$$x_2 \geq x_1^2 , \quad x_2 \leq x_1^2 . \tag{65}$$

Introduce the auxiliary variables x_3 and x_4 defined by

$$x_2 - x_1^2 - x_3^2 = 0, \quad x_1^2 - x_2 - x_4^2 = 0 . \tag{66}$$

Then, the previous problem can be recast as that of minimizing the

function (64) subject to the equality constraints (66). The function (64) admits the relative minimum $f = -1$ at the point defined by

$$x_1 = 1 , \quad x_2 = 1 , \quad x_3 = 0 , \quad x_4 = 0 \tag{67}$$

and

$$\lambda_1 = -1 , \quad \lambda_2 = -1 . \tag{68}$$

Example 8.8. Consider the problem of minimizing the function

$$f = \log x_3 - x_2 , \tag{69}$$

subject to the equality constraint

$$x_1^2 + x_3^2 - 4 = 0 , \tag{70}$$

and the inequality constraint

$$x_3 \geq 1 . \tag{71}$$

Introduce the auxiliary variable x_4 defined by

$$x_3 = 1 + x_4^2 . \tag{72}$$

Then, the previous problem can be recast as that of minimizing the function

$$f = \log(1 + x_4^2) - x_2 , \tag{73}$$

subject to the equality constraint

$$(1 + x_4^2)^2 + x_1^2 - 4 = 0 . \tag{74}$$

Note that x_3 has been eliminated from the problem and can be computed *a posteriori* with (72). The function (73) admits the relative minimum $f = -\sqrt{3}$ at the point defined by

$$x_1 = 0, \quad x_2 = \sqrt{3}, \quad x_4 = 1 \tag{75}$$

and

$$\lambda_1 = 1/2\sqrt{3} . \tag{76}$$

9. Results and Conclusions

The examples described in Section 8 were solved with the three versions of SGRA and the three versions of CGRA described in Section 3. The numerical results are presented in Tables 2-3, where the number of iterations for convergence N_ϵ is shown. For the eight examples considered, Table 4 shows the cumulative number of iterations for convergences ΣN_ϵ. From the tables, the following conclusions arise: (a) a restoration of some form is necessary for rapid convergence; and (b) while SGRA-CR is the most stable among the algorithms considered here, rapidity of convergence can be increased somewhat if one employs algorithms with alternate restoration or optional restoration.

Table 2. Number of iterations for convergence N_ϵ.

Example	SGRA-CR	SGRA-IR	SGRA-OR
8.1	5	5	5
8.2	8	8	8
8.3	18	14	16
8.4	56	51	42
8.5	8	7	7
8.6	15	12	16
8.7	9	15	9
8.8	11	11	10

Table 3. Number of iterations for convergence N_ϵ.

Example	CGRA-NR	CGRA-AR	CGRA-OR
8.1	17	5	5
8.2	65	8	8
8.3	22	16	16
8.4	36	54	43
8.5	7	7	7
8.6	>100	19	13
8.7	13	7	9
8.8	15	8	10

Table 4. Cumulative number of iterations for convergence ΣN_e.

Algorithm	ΣN_e
SGRA-CR	130
SGRA-IR	123
SGRA-OR	113
CGRA-NR	> 275
CGRA-AR	124
CGRA-OR	111

REFERENCES

1. Miele, A., Huang, H.Y., and Heideman, J.C., *Sequential Gradient-Restoration Algorithm for the Minimization of Constrained Functions, Ordinary and Conjugate Gradient Versions*, Journal of Optimization Theory and Applications, Vol.4, No.4, 1969.

2. Miele, A., and Heideman, J.C., *Mathematical Programming for Constrained Minimal Problems, Part I, Sequential Gradient-Restoration Algorithm*, Rice University, Aero-Astronautics Report No.59,1969.

3. Miele, A., Heideman, J.C., and Levy, A.V., *Mathematical Programming for Constrained Minimal Problems, Part 3, Combined Gradient-Restoration Algorithm*, Rice University, Aero-Astronautics Report No.68,1970.

This research was supported by the Office of Scientific Research, Office of Aerospace Research, United States Air Force, Grant No.AF-AFOSR-72-2185

Comparison of Multiplier and Quasilinearization Methods

Alejandro V. Levy
Antonio Montalvo

Reprinted from I&EC Process Design and Development
Volume 11, October 1972, Page

functions f and c with respect to x exist and are continuous; it is also assumed that the constrained minimum exists.

2.1. Exact First-Order Conditions. From theory of maxima and minima, it is known that the previous problem can be restated as that of minimizing the *augmented function*

$$F(x,\lambda) = f(x) + \lambda^T c(x) \tag{3}$$

subject to the constraint (2). Here, λ is a q-vector Lagrange multiplier and the superscript T denotes the transpose of a matrix. If

$$F_x(x,\lambda) = f_x(x) + c_x(x)\lambda \tag{4}$$

denotes the gradient of the augmented function (In eq 4, the gradients f_x and F_x denote n-vectors and the matrix c_x is $n \times q$.), the optimum solution x,λ must satisfy the simultaneous equations

$$c(x) = 0;\ F_x(x,\lambda) = 0 \tag{5}$$

2.2. Approximate Solutions. In general, the system (5) is nonlinear; consequently, approximate methods must be employed. These are of two kinds: first-order methods (see, for instance, Miele et al. (1965)) and second-order methods. Here, we introduce the scalar quantities

$$P(x) = c^T(x)c(x);\ Q(x,\lambda) = F_x^T(x,\lambda)F_x(x,\lambda) \tag{6}$$

which measure the errors in the constraint and the optimum condition, respectively. We observe that: $P = 0$ and $Q = 0$ for the optimum solution, while $P > 0$ and/or $Q > 0$ for any approximation to the solution. When approximate methods are used, they must ultimately lead to values of x,λ such that

$$P(x) \le \epsilon_1;\ Q(x,\lambda) \le \epsilon_2 \tag{7}$$

Alternately, (7) can be replaced by

$$R(x,\lambda) \le \epsilon_3 \tag{8}$$

where

$$R(x,\lambda) = P(x) + Q(x,\lambda) \tag{9}$$

denotes the cumulative error in the constraint and the optimum condition. Here, $\epsilon_1,\epsilon_2,\epsilon_3$ are small, preselected numbers. Note that satisfaction of inequality 8 implies satisfaction of inequalities 7. If one chooses $\epsilon_1 = \epsilon_2 = \epsilon_3$.

3. Review of the SQL-Algorithm

Here, a review of SQL is given. Let x,λ denote the nominal values, and let $\Delta x, \Delta\lambda$ denote the displacements leading from the nominal values to the varied values $\bar{x},\bar{\lambda}$. With this understanding, SQL can be summarized as follows.

(i) For given x and λ, compute $f(x)$, $c(x)$, $f_x(x)$, $c_x(x)$.

(ii) Compute

$$F_x(x,\lambda) = f_x(x) + c_x(x)\lambda \tag{10-1}$$

$$Q(x,\lambda) = F_x^T(x,\lambda)F_x(x,\lambda) \tag{10-2}$$

$$P(x) = c^T(x)c(x) \tag{10-3}$$

(iii) If $P(x) \le \epsilon_1$ and $Q(x,\lambda) \le \epsilon_2$, convergence is achieved and the algorithm is stopped; otherwise, go to step (iv).

(iv) Compute $f_{xx}(x)$, $c_{xx}(x)$ and obtain

$$F_{xx}(x,\lambda) = f_{xx}(x) + c_{xx}(x)\lambda \tag{11}$$

(v) Obtain the matrix

$$W(x,\lambda) = F_{xx}(x,\lambda) + k\,P_{xx}(x) \tag{12}$$

(vi) Obtain the varied point

$$\bar{x} = x + \Delta x \tag{13-1}$$

$$\bar{\lambda} = \lambda + \Delta\lambda \tag{13-2}$$

and go back to step (i).

4. Review of the MIMI-Algorithm

Here, a review of MIMI is given in conjunction with the modified quasi-linearization algorithm. Let x denote the nominal point, $\bar{x}$ the varied point, Δx the displacement leading from the nominal point to the varied point, p the search direction, $s = \pm 1$ the direction factor, and α the step size. With this understanding, the modified method of the multipliers can be summarized as follows.

(i) Choose a nominal value for x not satisfying the constraint. Compute $f(x)$, $c(x)$.

(ii) Compute $f_x(x)$, $c_x(x)$ and obtain

$$P(x) = c^T(x)c(x) \tag{14-1}$$

$$P_x(x) = 2c_x(x)c(x) \tag{14-2}$$

$$F_x(x,\lambda_1) = f_x(x) + c_x(x)\lambda_1 \tag{14-3}$$

$$\beta = -P_x^T(x)F_x(x,\lambda_1)/P_x^T(x)P_x(x) \tag{14-4}$$

where λ_1 is the Lagrange multiplier of the previous iteration.

(iii) Update the Lagrange multiplier to the value λ_2 given by

$$\lambda_2 = \lambda_1 + 2\beta c(x) \tag{15}$$

(iv) Compute

$$F_x(x,\lambda_2) = f_x(x) + c_x(x)\lambda_2 \tag{16-1}$$

$$Q(x,\lambda_2) = F_x^T(x,\lambda_2)F_x(x,\lambda_2) \tag{16-2}$$

(v) Obtain a tentative penalty constant

$$k_0 = |\lambda_2^T c(x)|/P(x) \tag{17-1}$$

and the updated penalty constant k_2 as follows

$$k_2 = \min(k_0,k_1) \text{ if } P(x) \le Q(x,\lambda_2) \tag{17-2}$$

$$k_2 = \max(k_0,k_1) \text{ if } P(x) > Q(x,\lambda_2) \tag{17-3}$$

where k_1 is the penalty constant of the previous iteration.

(vi) Compute the augmented penalty function and its gradient

$$W(x,\lambda_2,k_2) = f(x) + \lambda_2^T c(x) + k_2 P(x) \tag{18-1}$$

$$W_x(x,\lambda_2,k_2) = f_x(x) + c_x(x)\lambda_2 + k_2 P_x(x) \tag{18-2}$$

(vii) Compute the second derivative matrices $f_{xx}(x)$, $c_{xx}(x)$ and obtain

$$F_{xx}(x,\lambda_2) = f_{xx}(x) + c_{xx}(x)\lambda_2 \tag{19-1}$$

$$P_{xx}(x) = 2\,[c_{xx}(x)c(x) + c_x(x)c_x^T(x)] \tag{19-2}$$

$$W_{xx}(x,\lambda_2,k_2) = F_{xx}(x,\lambda_2) + k_2 P_{xx}(x) \tag{19-3}$$

(viii) Solve the linear system of equations of order n [illegible] for the search direction p [illegible]

$$W_{xx}(x,\lambda_2,k_2)p + W_x(x,\lambda_2,k_2) = 0 \tag{20}$$

(ix) Set the stepsize at the value $\alpha = 1$ and compute the direction factor s and the search direction p from the relations

$$p = \rho\lambda \tag{21-2}$$

(x) Compute the displacement Δx, the varied point $\bar{x}$, and the new value of the augmented penalty function with

$$\Delta x = -\alpha p \tag{22-1}$$

$$\bar{x} = x + \Delta x \tag{22-2}$$

$$W(\bar{x},\lambda_2,k_2) = f(\bar{x}) + \lambda_2^T\varphi(\bar{x}) + k_2P(\bar{x}) \tag{22-3}$$

(xi) If

$$W(\bar{x},\lambda_2,k_2) < W(x,\lambda_2,k_2) \tag{22-4}$$

accept the present value of the step size α and go back to step (ii); otherwise, go to step (xii).

(xii) If

$$W(\bar{x},\lambda_2,k_2) > W(x,\lambda_2,k_2) \tag{22-5}$$

bisect the step size as many times as needed (The bisections are started from $\alpha = 1.0$) and go back to step (x) until inequality (22-4) is satisfied.

Remark. For the first iteration, the previous values of λ_2 and k_2 are not known. This being the case, one sets

$$\lambda_2 = 0; k_2 = k_0 \tag{23}$$

5. Operational Count

In this section, we give the number of arithmetic operations performed by each algorithm per iteration. This operational count gives a measure of the computer time needed per iteration. The computer time needed to compute the functions $f(x)$ and $\varphi(x)$ and the derivatives $f_x(x)$, $\varphi_x(x)$, $f_{xx}(x)$, $\varphi_{xx}(x)$ is not considered in the comparison. In the absence of step size bisections, SQL and MMM require the same number of function evaluations per iteration. Therefore, the number of function evaluations is not important for the comparison if the basic criterion is the computer time difference between the two algorithms.

We note that the summation and subtraction operations require much less computer time than the multiplication and division operations. Therefore, the operational count is done only on the basis of the number of multiplications and divisions required to complete one iteration.

5.1. Operational Count for the SQL-Algorithm. Let μ_1 denote the number of multiplications and divisions required by the SQL-algorithm per iteration in order to solve the linear system of equations of order $n + q$ (Isaacson and Keller (1966)). Let μ_2 denote the number of remaining operations. These numbers are given by

$$\mu_1 = (n + q)^3/3 + (n + q)^2 - (n + q)/3 \tag{24-1}$$

$$\mu_2 = n^2q/2 + 3nq/2 + n + q \tag{24-2}$$

As a consequence, their sum

$$\mu = \mu_1 + \mu_2 \tag{25}$$

becomes

$$\mu = (n + q)^3/3 + (n + q)^2 - (n + q)/3 + n^2q/2 + 3nq/2 + n + q \tag{26}$$

[illegible] Let μ_1 denote the number of multiplications and divisions required by the MMM-algorithm per iteration in order to solve the linear system of equations of order n (Isaacson and Keller (1966)). Let μ_2 denote the number of remaining operations. These numbers are given by

$$\mu_1 = n^3/3 + n^2 - n/3 \tag{27}$$

n	q_c	n	q_c	n	q_c
2	2	20	5	200	14
3	3	30	6	300	17
4	3	40	7	400	20
5	3	50	7	500	22
6	3	60	8	600	24
7	4	70	9	700	26
8	4	80	9	800	28
9	4	90	10	900	30
10	4	100	10	1000	32

$$\mu_2 = 3n^2q/2 + 9nq/2 + 3(n + q) + n^2 + 6n + 2q + 5 \tag{25}$$

As a consequence, their sum (25) becomes

$$\mu = n^3/3 + 3n^2q/2 + 2n^2 + 9nq/2 + 26n/3 + 5q + 5 \tag{26}$$

5.2. Comparison of Operational Counts. Comparison between SQL and MMM shows that the former requires a larger number of operations to solve the linear system and a smaller number of operations to perform the remaining tasks. It is of interest to compute the difference Δ between the overall number of operations, that is

$$\Delta = (\mu)_{SQL} - (\mu)_{MMM} \tag{30}$$

From eq 26, 29, and 30, we obtain

$$\Delta = q^3/3 + q^2(n + 1) - (n^2 + nq + 8n + 13q/3 + 5) \tag{31}$$

Now, let $\Delta = 0$, so that eq 31 becomes

$$q^3/3 + q^2(n + 1) - (n^2 + nq + 8n + 13q/3 + 5) = 0 \tag{32}$$

For given n, let q_r denote the solution of eq 32 and let q_c denote the closest integer (from below) to q_r, subject to the limitation $q_c \leq n$. Inspection of eq 31 and 32 shows that, for any given n

$$\Delta \leq 0; 0 \leq q \leq q_c \tag{33-1}$$

$$\Delta > 0; q_c + 1 \leq q \leq n \tag{33-2}$$

The values of q_c are given in Table I as a function of the number of variables n. As an example, consider the case $n = 10$, for which $q_c = 4$. If $q \leq 4$, SQL requires a smaller number of operations than MMM. On the other hand, if $q \geq 5$, the opposite is true. From the table, it appears that MMM might become superior to SQL for large systems (large n), even when these large systems are moderately constrained.

6. Experimental Conditions

In order to illustrate the characteristics of the SQL and MMM algorithms, nine numerical examples were solved using an IBM 370/155 computer and double precision arithmetic. Both algorithms were programmed in Fortran [illegible]

Nominal Values. For both algorithms, the nominal [illegible] following values of d

[illegible] (34)

For the SQL-algorithm, the nominal values of the Lagrange

multipliers were chosen as $\lambda_i = \gamma, i = 1,2,3,\ldots,q$, with the following values of γ

$$\gamma = \pm 10, \pm 5, \pm 0, \pm 1, \pm 2 \qquad (34\text{-}2)$$

For the MMM-algorithm, one starts with $\lambda = 0$ (section 4). Therefore, 100 runs were made for each problem with the SQL-algorithm, and 10 runs were made for each problem with the MMM-algorithm.

Stopping Conditions. The convergence criterion for both algorithms was chosen to be

$$P(x) + Q(x,\lambda) \leq 10^{-6} \qquad (35)$$

Conversely, the nonconvergence criteria were chosen as follows:

$$\text{(a) } N \geq 100 \qquad (35\text{-}1)$$

$$\text{(b) } N_b \geq 20 \qquad (35\text{-}2)$$

$$\text{(c) } M \geq 10^{75} \qquad (35\text{-}3)$$

Here, N is the iteration number, N_b is the number of bisections of the stepsize α required to satisfy inequality (22-4), and M is the modulus of any of the quantities employed in the algorithm. Condition (a) implies slow convergence; condition (b) denotes extreme smallness of the displacements and, consequently, slow convergence; and, finally, condition (c) denotes an overflow in the number range of the computer.

Percentage of Success. Each problem was solved several times starting with the nominal values given by eq 34. Let N_r denote the total number of runs made for a given problem, using a given method; let N_s denote the number of runs for which the algorithm succeeded in converging to a relative minimum. Then, the percentage of success p is given by

$$p = N_s/N_r \qquad (37)$$

It is noted that $0 \leq p \leq 1$ and that the higher the value of p, the more robust the algorithm is, since it means that it has a larger range of successful convergence.

Actual Computer Time per Run. Let T_i denote the CPU time in seconds employed in the ith run to solve a given problem with a given algorithm. For a given problem and a given algorithm, the average CPU time per successful run is given by

$$T_{av} = (1/N_s)\sum_{i=1}^{N_r} T_i \qquad (38)$$

Effective Computer Time per Run. The main characteristics of the algorithms considered in this report, namely, robustness (high p) and speed (low T_{av}) can be combined in a single parameter by considering the effective CPU time per run, which, for a given problem and a given algorithm, is given by

$$T_{eff} = T_{av}/p \qquad (39)$$

Thus, an algorithm which is fast and has a high percentage of success has a smaller effective computer time than an algorithm which is slow and has a small percentage of success.

Relative Efficiency of Two Algorithms. In order to compare the SQL- and MMM-algorithms and arrive at a significant parameter which is machine independent, we introduce the [illegible] efficiency index

$$E = (T_{eff})_{SQL}/(T_{eff})_{MMM} \qquad (40)$$

Therefore, $E < 1$ implies that SQL is more efficient than MMM, while the opposite is true for $E > 1$.

7. Numerical Examples

In this section nine numerical examples are described. Examples 7.1 through 7.7 were considered by Miele et al. (1972), while example 7.8 was considered by Lootsma and Jaakola (1973). For simplicity, scalar notation is used.

Example 7.1. Consider the problem of minimizing the function

$$f = (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2 \qquad (41)$$

subject to the constraints

$$x_1 + 3x_2 = 0;\ x_3 + x_4 - 2x_5 = 0;\ x_2 - x_5 = 0 \qquad (42)$$

The function (41) admits the relative minimum $f = 0.4093 E + 01$ at the point defined by

$$x_1 = -0.7674;\ x_2 = 0.2558;\ x_3 = 0.6279;$$
$$x_4 = -0.1162;\ x_5 = 0.2558 \qquad (43)$$

$$\lambda_1 = 0.2046 E + 01;\ \lambda_2 = 0.2222 E + 01;$$
$$\lambda_3 = -0.5953 E + 01 \qquad (44)$$

Example 7.2. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^4 \qquad (45)$$

subject to the constraint

$$x_1(1 + x_2^2) + x_3^4 - 4 - 3\sqrt{2} = 0 \qquad (46)$$

(i) At the point defined by

$$x_1 = 0.1104 E + 01;\ x_2 = 0.1196 E + 01;$$
$$x_3 = 0.1535 E + 01 \qquad (47)$$

$$\lambda_1 = -0.1072 E - 01 \qquad (48)$$

the function (45) has the relative minimum $f = 0.8250 E - 01$.

(ii) At the point defined by

$$x_1 = 0.9861 E - 01;\ x_2 = -0.8954;\ x_3 = -0.1665 E + 01 \qquad (49)$$

$$\lambda_1 = -0.1029 \qquad (50)$$

the function (45) has the relative minimum $f = 0.2199 E + 01$.

Example 7.3. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_3 - 1)^2 + (x_4 - 1)^4 + (x_5 - 1)^6 \qquad (51)$$

subject to the constraints

$$x_1^2 x_4 + \sin(x_4 - x_5) - 2\sqrt{2} = 0;$$
$$x_2 + x_3^4 x_4^2 - 8 - \sqrt{2} = 0 \qquad (52)$$

(i) At the point defined by

$$x_1 = 0.1166 E + 01;$$
$$x_2 = 0.1152 E + 01;$$
$$x_3 = 0.1092 E + 01;$$
$$x_4 = 0.1066 E + 01;\ x_5 = 0.6103 \qquad (53)$$

and

$$\lambda_1 = -0.5753 E - 01;\ \lambda_2 = -0.3157 E - 01 \qquad (54)$$

the function (51) has the relative minimum $f = 0.2415$.

(ii) At the point defined by

$x_1 = 0.1059\ E + 01;\ x_2 = 0.1177\ E + 01;$

$x_3 = -0.1231\ E + 01;\ x_4 = 0.1747\ E + 01;$

$x_5 = 0.5213$ (55)

and

$$\lambda_1 = -0.1469\ E - 03;\ \lambda_2 = -0.1774 \quad (56)$$

the function (51) has the relative minimum $f = 0.5333\ E + 01$.

(iii) At the point defined by

$x_1 = -0.1028\ E + 01;\ x_2 = -0.1017\ E + 01;$

$x_3 = 0.1354\ E + 01;\ x_4 = 0.1760;\ x_5 = 0.4531$ (57)

and

$$\lambda_1 = -0.1126\ E + 01;\ \lambda_2 = -0.2301\ E - 01 \quad (58)$$

the function (51) has the relative minimum $f = 0.4002\ E + 01$.

(iv) At the point defined by

$x_1 = -0.9368;\ x_2 = -0.9142;\ x_3 = -0.1802\ E + 01;$

$x_4 = 0.1803\ E + 01;\ x_5 = 0.4975$ (59)

and

$$\lambda_1 = -0.1102\ E + 01;\ \lambda_2 = -0.1452 \quad (60)$$

the function (51) has the relative minimum $f = 0.9008\ E + 01$.

Example 7.4. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + (x_3 - x_4)^4 + (x_4 - x_5)^4 \quad (61)$$

subject to the constraints

$$x_1 + x_2^2 + x_3^3 - 2 - 3\sqrt{2} = 0;$$

$$x_2 - x_3^2 + x_4 + 2 - 2\sqrt{2} = 0;\ x_1 x_5 - 2 = 0 \quad (62)$$

(i) At the point defined by

$x_1 = 0.1191\ E + 01;\ x_2 = 0.1362\ E + 01;$

$x_3 = 0.1472\ E + 01;$

$x_4 = 0.1635\ E + 01;\ x_5 = 0.1679\ E + 01$ (63)

and

$\lambda_1 = -0.3882\ E - 01;\ \lambda_2 = -0.1672\ E - 01;$

$\lambda_3 = -0.2873\ E - 03$ (64)

the function (61) has the relative minimum $f = 0.7977\ E - 01$.

(ii) At the point defined by

$x_1 = 0.2717\ E + 01;\ x_2 = 0.2033\ E + 01;\ x_3 = -0.8479;$

$x_4 = -0.4859;\ x_5 = 0.7359$ (65)

and

$\lambda_1 = -0.2823\ E + 01;\ \lambda_2 = 0.7106\ E + 01;$

$\lambda_3 = -0.2684\ E + 01$ (66)

the function (61) has the relative minimum $f = 0.1396\ E + 02$.

(iii) At the point defined by

$x_1 = -0.7934;\ x_2 = 0.2046\ E + 01;\ x_3 = -0.4681;$

$x_4 = -0.1102\ E + 01;\ x_5 = -0.4834\ E + 01$ (67)

and

$\lambda_1 = -0.2575\ E + 01;\ \lambda_2 = 0.2202\ E + 01;$

$\lambda_3 = -0.5085\ E + 01$ (68)

the function (61) has the relative minimum $f = 0.2745\ E + 02$.

(iv) At the point defined by

$x_1 = -0.1246\ E + 01;\ x_2 = 0.2122\ E + 01;$

$x_3 = 0.1174\ E + 01;\ x_4 = -0.2132;$

$x_5 = -0.1604\ E + 01$ (69)

and

$\lambda_1 = -0.2016\ E + 01;\ \lambda_2 = -0.6231\ E - 01;$

$\lambda_3 = -0.8632\ E + 01$ (70)

the function (61) has the relative minimum $f = 0.2752\ E + 02$.

(v) At the point defined by

$x_1 = 0.9494;\ x_2 = -0.2296\ E + 01;\ x_3 = 0.5377;$

$x_4 = 0.3384\ E + 01;\ x_5 = 0.2106\ E + 01$ (71)

and

$\lambda_1 = -0.2484\ E + 02;\ \lambda_2 = -0.1006\ E + 03;$

$\lambda_3 = 0.8790\ E + 01$ (72)

the function (61) has the relative minimum $f = 0.8652\ E + 02$.

(vi) At the point defined by

$x_1 = -0.2702\ E + 01;\ x_2 = -0.2989\ E + 01;\ x_3 = 0.1719;$

$x_4 = 0.3817\ E + 01;\ x_5 = -0.7401$ (73)

and

$\lambda_1 = -0.9898\ E + 02;\ \lambda_2 = -0.5950\ E + 03;$

$\lambda_3 = -0.1429\ E + 03$ (74)

the function (61) has the relative minimum $f = 0.6495\ E + 03$.

Example 7.5. Consider the problem of minimizing the function

$$f = 0.01(x_1 - 1)^2 + (x_2 - x_1^2)^2 \quad (75)$$

subject to constraint

$$x_1 + x_3^2 + 1 = 0 \quad (76)$$

At the point defined by

$$x_1 = -1;\ x_2 = 1;\ x_3 = 0 \quad (77)$$

$$\lambda_1 = 0.04 \quad (78)$$

this function (75) has the relative minimum $f = 0.04$.

Example 7.6. Consider the problem of minimizing the function

$$f = -x_3 \quad (79)$$

subject to constraints

$$x_2 - x_1^3 - x_3^2 = 0;\ x_1^2 - x_2 - x_4^2 = 0 \quad (80)$$

The function (79) admits the relative minimum $f = -1$ at the point defined by

$$x_1 = 1;\ x_2 = 1;\ x_3 = 0;\ x_4 = 0 \quad (81)$$

$$\lambda_1 = -1;\ \lambda_2 = -1 \quad (82)$$

Example 7.7. Consider the problem of minimizing the function

$$f = \log(1 + x_1^2) - x_2 \quad (83)$$

subject to the constraint

$$(1 - x_1)^3 + x_2^3 - 4 = 0 \quad (84)$$

(i) At the point defined by

Table II. Results for the Examples

	SQL-algorithm			MMM-algorithm		
Example	N_r	N_s	$\sum_{i=1}^{N_s} t_i$	N_r	N_s	$\sum_{i=1}^{N_s} t_i$
7.1	100	100	3.80	10	10	1.55
7.2	100	95	12.02	10	9	1.02
7.3	100	91	67.45	10	8	5.27
7.4	100	78	93.32	10	10	3.03
7.5	100	100	13.89	10	7	1.56
7.6	100	90	27.79	10	10	3.07
7.7	100	51	4.45	10	10	1.17
7.8	100	19	1.67	10	10	1.15
7.9	100	22	50.72	10	10	27.70

Table III. Results for the Examples

Example	SQL-algorithm			MMM-algorithm			
	p	T_{av}	T_{act}	p	T_{av}	T_{act}	E
7.1	1.00	0.058	0.058	1.00	0.133	0.133	0.43
7.2	0.95	0.136	0.143	0.90	0.113	0.125	1.14
7.3	0.91	0.743	0.816	0.80	0.658	0.823	0.99
7.4	0.78	0.427	0.547	1.00	0.303	0.303	1.00
7.5	1.00	0.135	0.135	0.70	0.220	0.314	0.42
7.6	0.90	0.308	0.342	1.00	0.307	0.307	1.11
7.7	0.51	0.087	0.170	1.00	0.117	0.117	1.45
7.8	0.19	0.103	0.545	1.00	0.115	0.115	4.73
7.9	0.22	2.300	20.450	1.00	2.771	2.771	3.77

$$x_1 = 0; x_2 = \sqrt{3} \tag{85}$$

$$\lambda_1 = \sqrt{3}/6 \tag{86}$$

the function (83) has the relative minimum $f = -\sqrt{3}$.

(ii) At the point defined by

$$x_1 = 0; x_2 = \sqrt{3} \tag{87}$$

$$\lambda_1 = -\sqrt{3}/6 \tag{88}$$

the function (83) has the relative minimum $f = \sqrt{3}$.

Example 7.8. Consider the problem of minimizing the function

$$f = -(x_1^2 + x_2^2 + x_3^2) \tag{89}$$

subject to the constraints

$$x_1 + 2x_2 + 3x_3 - 1 = 0; x_1^2 + x_2^2/2 + x_3^2/4 - 4 = 0 \tag{90}$$

(i) At the point defined by

$$x_1 = -0.1535; x_2 = -0.2429\,E+01; x_3 = 0.2014\,E+01 \tag{91}$$

$$\lambda_1 = 0.5960; \lambda_2 = 0.2435\,E+01 \tag{92}$$

the function (89) has the relative minimum $f = -0.9565\,E+01$.

(ii) At the point defined by

$$x_1 = 0.3442; x_2 = 0.2022\,E+01; x_3 = -0.1455\,E+01$$

$$\lambda_1 = -0.4121; \lambda_2 = 0.2314\,E+01 \tag{93}$$

the function (89) has the relative minimum $f = -0.9556\,E+01$.

Example 7.9. Consider the problem of minimizing the function

$$f = -\sum_{i=1}^{10} x_i^2 \tag{94}$$

subject to the constraints

$$\sum_{i=1}^{10} (x_i/c_{ij}) - 1 = 0; j = 1, 2, \ldots, 9 \tag{96}$$

$$\sum_{i=1}^{10} (x_i^2/a_i^2) - 4 = 0 \tag{97}$$

where the coefficients c_{ij} and a_i are given by

$$c_{ij} = 1; i \neq j$$
$$c_{ij} = 2; i = j \tag{98}$$

and

$$a_i^2 = 1 + 3(i-1)/9; i = 1, 2, \ldots, 10 \tag{99}$$

(i) At the point defined by

$$x_1 = -0.9158\,E-01; x_2 = -0.9158\,E-01;$$
$$x_3 = -0.9158\,E-01; x_4 = -0.9158\,E-01;$$
$$x_5 = -0.9158\,E-01; x_6 = -0.9158\,E-01;$$
$$x_7 = -0.9158\,E-01;$$
$$x_8 = -0.9158\,E-01; x_9 = -0.1614\,E+01;$$
$$x_{10} = 0.3501\,E+01 \tag{100}$$

and

$$\lambda_1 = -0.6332; \lambda_2 = -0.2779; \lambda_3 = -0.6480\,E-01;$$
$$\lambda_4 = 0.7732\,E-01; \lambda_5 = 0.1756; \lambda_6 = 0.2549;$$
$$\lambda_7 = 0.3141; \lambda_8 = 0.3615; \lambda_9 = 0.3879\,E+01 \tag{101}$$

the function (95) has the relative minimum $f = -0.1562\,E+01$.

(ii) At the point defined by

$$x_1 = 0.1064; x_2 = 0.1064; x_3 = 0.1064;$$
$$x_4 = 0.1064; x_5 = 0.1064; x_6 = 0.1064;$$
$$x_7 = 0.1064; x_8 = 0.1064; x_9 = 0.2843\,E+01;$$
$$x_{10} = -0.2842\,E+01 \tag{102}$$

and

$$\lambda_1 = 0.7234; \lambda_2 = 0.5156; \lambda_3 = 0.7462\,E-01;$$
$$\lambda_4 = -0.9505\,E-01; \lambda_5 = -0.2042; \lambda_6 = -0.2914;$$
$$\lambda_7 = -0.3592; \lambda_8 = -0.4134; \lambda_9 = 0.3819\,E+01 \tag{103}$$

the function (95) has the relative minimum $f = -0.1516\,E+01$.

8. Numerical Results and Conclusions

The examples described in section 7 were solved using both the standard quasilinearization algorithm (SQL) and the modified method of multipliers (MMM) in accordance with the experimental conditions outlined in section 6.

Table II shows, for each example and each algorithm, the total number of runs N_r, the number of successful runs N_s, and the [illegible] N_s runs. Table III shows the [illegible] and the [illegible] T_{av} and the [illegible] derived.

[illegible] examples [illegible] lower than 70% for MMM. On the other hand, for SQL, the

and 20% in example 7.9.

On the upper side, MMM achieved a percentage of success of 100% in six examples, while SQL achieved a percentage of success of 100% in two examples (one of which is the obvious linear-quadratic example 7.1).

For the nine examples, the cumulative percentage of success was 93% for MMM and 74% for SQL. From all these data, the higher robustness of MMM is apparent.

(ii) Computer Time per Run. Inspection of the average computer time per run T_{av} shows that SQL is superior to MMM in five examples and inferior in four examples. However, when one looks at the effective computer time per run T_{ef}, the situation is just the opposite: MMM is superior to SQL in six examples and inferior in three examples.

(iii) Relative Efficiency Index. In section 6, the relative efficiency index E was introduced as a way of combining percentage of success with average computer time per run, while arriving at a parameter which is machine independent, to some degree. This efficiency index E was defined as follows

$$E = \frac{(T_{ef})_{SQL}}{(T_{ef})_{MMM}} = \frac{(T_{av})_{SQL}}{(T_{av})_{MMM}} \frac{(p)_{MMM}}{(p)_{SQL}} \qquad (104)$$

This relative efficiency index is defined so that $E < 1$ indicates superiority of SQL with respect to MMM, while $E > 1$ indicates inferiority of SQL with respect to MMM. Inspection of Table III shows that $E < 1$ in three examples and $E > 1$ in six examples. Thus, from the examples investigated, we conclude that MMM compares favorably with SQL.

(iv) The results of Tables II and III must be taken with a grain of salt, since computer times are precise only to ±20%. This being the case, values of E in the range $0.8 \leq E$ ≤ 1.2 in four examples. Therefore, even accounting for possible imprecision in computer time measurements, MMM compares favorably with SQL.

(v) As shown by eq 31 and 32 and Table I, the relative advantage of MMM with respect to SQL should become more apparent for large systems (large n) involving many constraints (large q). Example 7.9, which includes $n = 10$ variables and $q = 9$ constraints supports this point of view (see Table III).

Acknowledgment

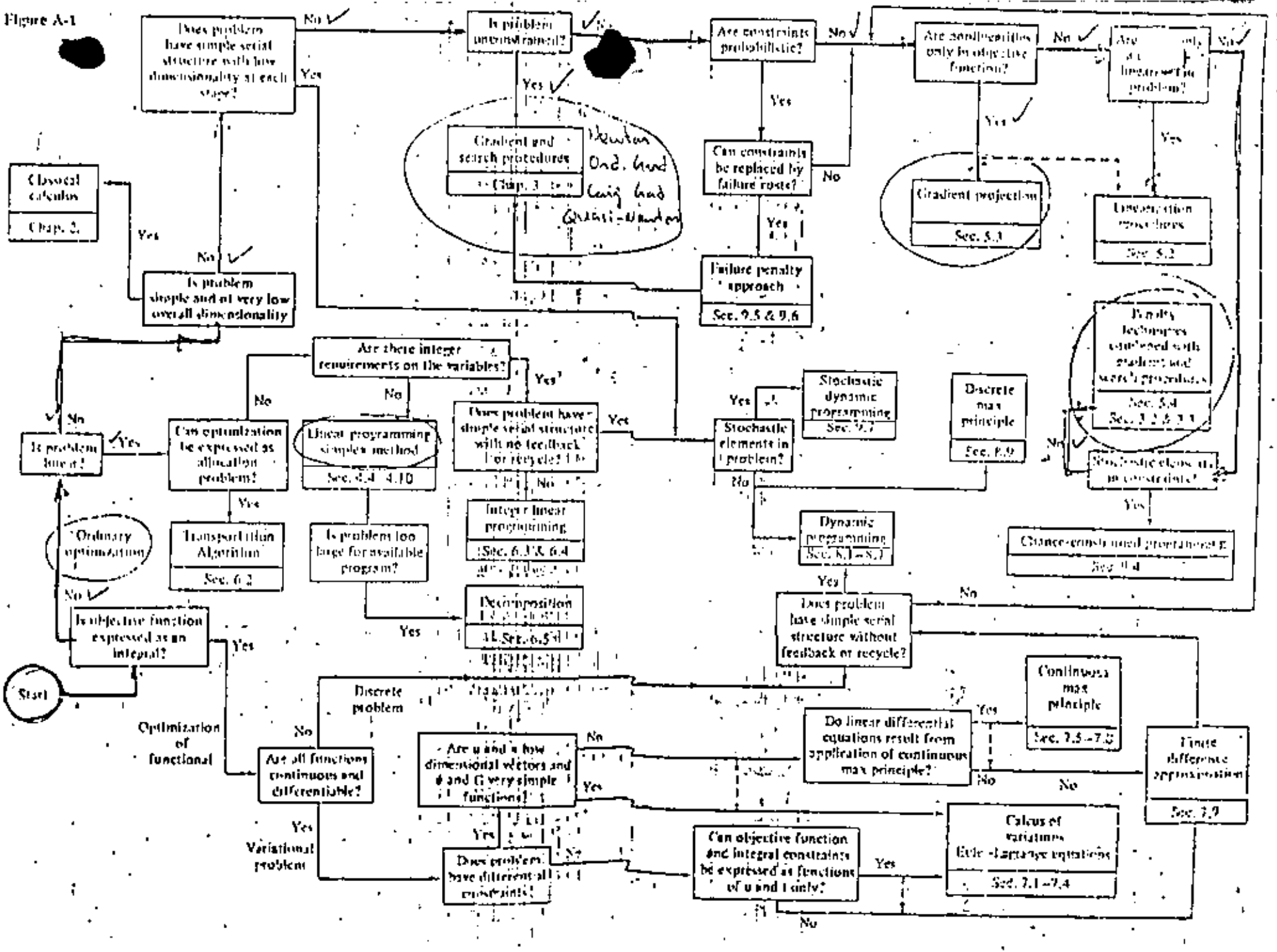
This research was supported by the National Science Foundation, Grant No. GP-41158. The authors are indebted to Professor A. Miele for stimulating discussions. This work was done while the first author held a post-doctoral position with the Aero-Astronautics Group of Rice University, Houston, Texas.

Literature Cited

Hestenes, M. R., *J. Optim. Theory Appl.* 4 (5), 303–320 (1969).
Isaacson, E., Keller, H. B., "Analysis of Numerical Methods," pp 34–37, Wiley, New York, N.Y., 1966.
Luus, R., Jaakola, H. E. Inf. *Eng. Chem., Process Des. Dev.*, 12, 380–383 (1973).
Miele, A., Huang, H. Y., Heideman, J. L., *J. Optim. Theory Appl.*, 4, (4), 213–243 (1969).
Miele, A., Levy, A. V., Iyer, R. R., Well, K. H., "Modified Quasilinearization Method for Mathematical Programming Problems and Optimal Control Problems," "Control and Dynamic Systems, Advances in Theory and Applications," Vol. 9, pp 239–307, C. T. Leondes, Ed., Academic Press, New York, N.Y., 1973.
Miele, A., Moseley, P. E., Levy, A. V., Coggins, G. M., *J. Optim. Theory Appl.*, 10, (1), 1–30 (1972).

Received for review August 26, 1974
Accepted May 14, 1975

Figure A-1



536

Reference: Introduction to Optimization Theory - B.S. Gottfried and J. Weisman

537

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

OPTIMACION DE SISTEMAS FISICOS

ESTUDIO DE CASOS:

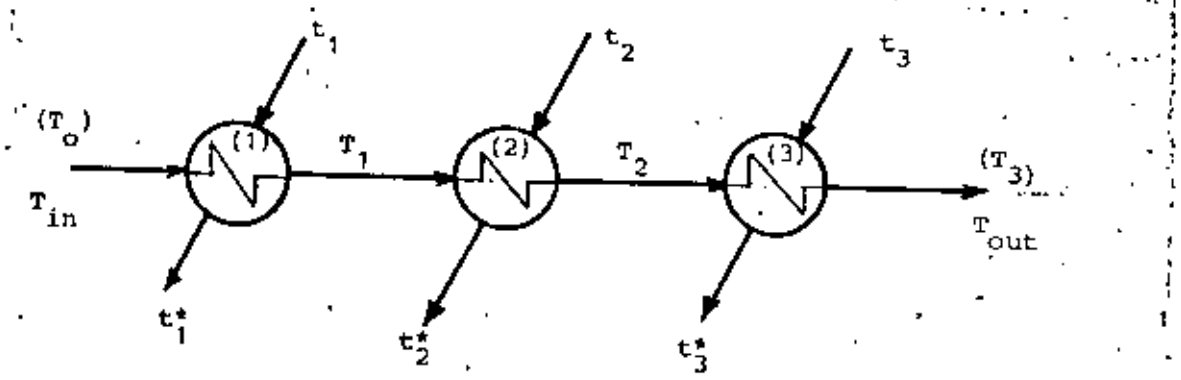
- Optimación de un tren de cambiadores de calor
- Ampliación de una planta química

M. EN C. SUSANA GOMEZ GOMEZ

MARZO, 1981

OPTIMIZACION DE UN TREN DE CAMBIADORES DE CALOR

Se desea calentar un fluido desde una temperatura T_{in} hasta una temperatura de salida T_{out}, mediante intercambio de calor con tres corrientes liquidas calientes, en un sistema de tres cambiadores de calor que operan en contra corriente, según se muestra en la figura



Para el proceso anterior se especifican los siguientes parámetros:

WC_q : producto del flujo en masa por la capacidad calorifica (se supone idéntica en todas las corrientes)

t_i : temperatura de entrada de las corrientes calientes ($i = 1, 2, 3$)

U_i : coeficiente total de transferencia de calor en cada uno de los tres cambiadores ($i = 1, 2, 3$)

Si se supone que la inversión total requerida para el sistema de cambiadores es proporcional al area total de los mismos, el problema se puede plantear como el de seleccionar las areas A_i $(i = 1,2,3)$ de manera

tal que :

$$A_T = A_1 + A_2 + A_3$$

sea mínima.

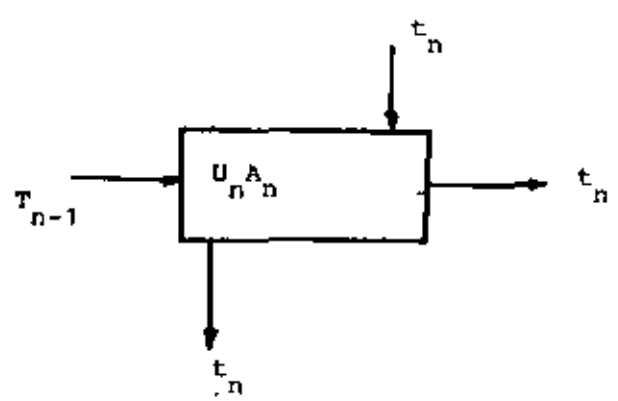
1°) Formular el problema como una optimización en estado estacionario. Identifique la función objetivo, variables de decisión y restricciones. Los siguientes parámetros se consideran fijos

Tin = To = 100	t_1 = 300	U_1 = 120
Tout = T_3 = 500	t_2 = 400	U_2 = 80
wc = 100,000	t_3 = 600	U_3 = 40

2°) Suponga que en el problema anterior las corrientes de salida caliente de los cambiadores 1 y 2 deben mezclarse y la temperatura resultante no deberá exceder los 230° . Plantee este nuevo problema tomando en cuenta la restricción planteada.

Modelo del cambiador de calor

Considerése el cambiador de calor a contra corriente que se muestra a continuación



Las ecuaciones que describen su funcionamiento son :

Q_n : rapidez de transferencia de calor en el n-ésimo cambiador

$$= WC\ (T_n - T_{n-1})$$

$$= WC\ (t_n - t_n^*)$$

$$= U_n A_n\ (t_n - T_n)$$

$$= U_n A_n\ (t_n^* - T_{n-1})$$

siendo las temperaturas de salida las siguientes :

$$T_n = \frac{T_{n-1} + \alpha_n \phi_n}{1 + \alpha_n}$$

donde $\alpha n = \dfrac{U_n A_n}{WC}$

y $t_n^* = t_n - (T_n - T_{n-1})$

Función objetivo

Según se mencionó con anterioridad, se trata de minimizar el arca total del sistema

$$A_T = A_1 + A_2 + A_3$$

Variables de decisión y restricciones

1- Sin restricciones en el mezclado (Parte 1)

a) T_1 y T_2 como variables de decisión

$$Q_1 = W\,C\,(T_1 - T_o) \qquad A_1 = Q_1/U_1\,(\phi_1 - T_1)$$

$$Q_2 = W\,C\,(T_2 - T_1) \qquad A_2 = Q_2/U_2\,(\phi_2 - T_2)$$

$$Q_3 = W\,C\,(T_3 - T_2) \qquad A_3 = Q_3/U_3\,(\phi_3 - T_3)$$

Restricciones sobre las variables independientes

$$T_o \leq T_1 \leq t_1$$

$$T_1 \leq T_2 \leq t_2$$

Restricciones sobre las variables dependientes

$$Q_i \geq 0 \qquad i = 1, 2, 3$$

o equivalentemente

$$A_i \geq 0 \qquad i = 1, 2, 3$$

b) Arcas A_1 y A_2 como variables independientes

$$T_1 = \frac{T_o + \alpha_1\, t_1}{1 + \alpha_1} \quad ; \quad \alpha_1 = U_1\, A_1/W\,C$$

$$T_2 = \frac{T_1 + \alpha_2\, t_2}{1 + \alpha_2} \quad ; \quad \alpha_2 = U_2\, A_2/W\,C$$

Q_1, Q_2 y Q_3 como en el caso anterior, al igual que A_3

restricciones en las variables independientes

$$0 \leq A_1 \leq A_1^* \quad \text{(dato)}$$

$$0 \leq A_2 \leq A_2^* \quad \text{(dato)}$$

restricciones en las variables dependientes : no hay.

c) Cargas térmicas Q_1 y Q_2 como variables de decisión

$$T_1 = Q_1 / W C + T_o \quad ; \quad A_1 = Q_1 / U_1 \ (t_1 - T_1)$$

$$T_2 = Q_2 / W C + T_1 \quad ; \quad A_2 = Q_2 / U_2 \ (t_2 - T_2)$$

$$Q_3 = W C \ (T_3 - T_2) \quad ; \quad A_3 = Q_3 / U_3 \ (t_3 - T_3)$$

Restricciones sobre las variables independientes

$$0 \leq Q_1 \leq W C \ (t_2 - T_o)$$

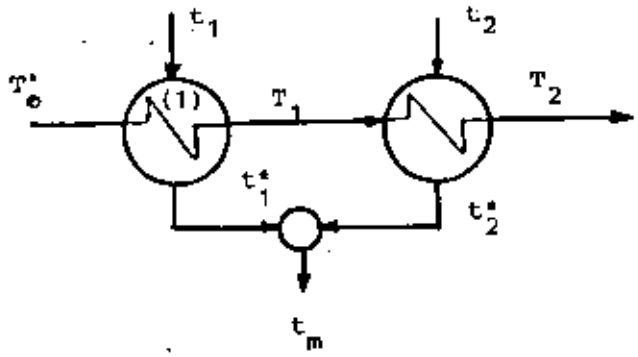
$$0 \leq Q_2 \leq W C \ (t_2 - T_o)$$

Restricciones sobre las variables dependientes

$$0 \leq Q_1 + Q_2 \leq W C \ (t_2 - T_o)$$

2- Con restricciones en la corriente de mezcla (Parte).

Las ecuaciones y restricciones analizadas en la parte 1 siguen siendo válidas aunque en este caso existe la necesidad de añadir otra restricción, según se analiza a continuación



ariable dependiente : $t_m^* = \dfrac{t_1^* + t_2^*}{2}$

restricciones sobre t_m :

λ Si $A_1 = \infty$ y $A_2 = \infty$ $\Longrightarrow$ $t_1^* = T_o$, $T_1 = t_1$

$t_2^* = T_1$, $T_2 = t_2$

entonces $\dfrac{t_1 + T_o}{2} \leq t_m^*$

y por lo tanto $\dfrac{t_1 + T_o}{2} \quad t_m^* \quad 230$ (restricción del problema)

↑ (restricción por arca infinita en A_1 y A_2)

Las soluciones óptimas resultan ser :

Primera parte Sin restricción en t_m^*

$T_1 = 186.2$, $T_2 = 292.7$, $Q_1 = 8.62 * 10^6$, $Q_2 = 10.64 * 10^6$, $Q_3 = 20.73 * 10^6$

$t_m^* = 631.8$, $A_2 = 1239.1$, $A_3 = 5183.8$, $A_T = 7054.8$

Segunda parte Con restricción en t_m^*

$T_1 = 210.0$, $T_2 = 340.1$, $Q_1 = 11.00 * 10^8$, $Q_2 = 13.01 * 10^6$, $Q_3 = 16.01 * 10^6$

$t_m^* = 229.9$, $A_1 = 1017.9$, $A_2 = 2715.7$, $A_3 = 4009.3$, $A_T = 7743.0$

AMPLIACION DE UNA PLANTA QUIMICA

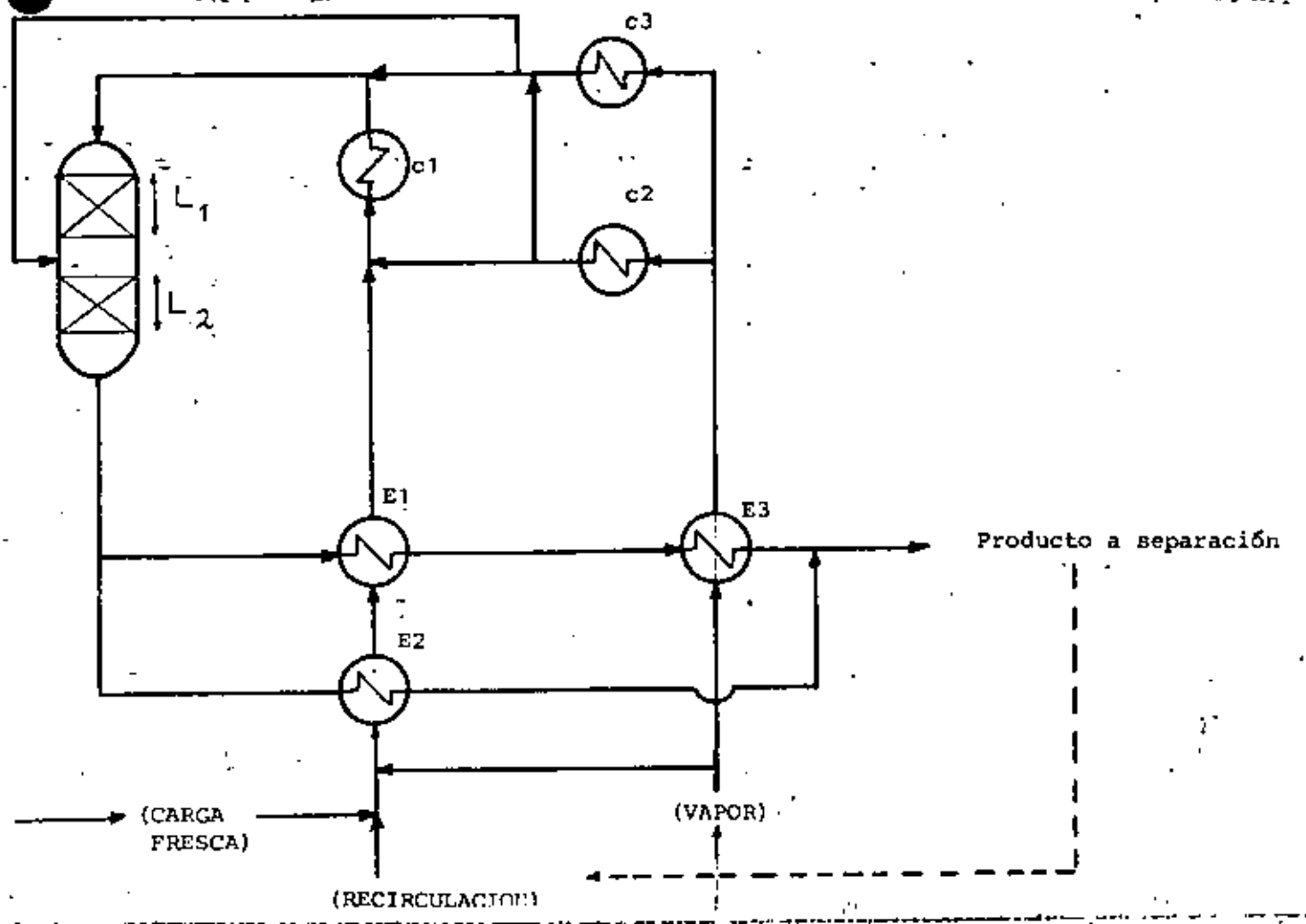
El presente estudio tiene el propósito de diseñar un nuevo reactor catalítico y fijar nuevas condiciones de operación para aumentar la capacidad de una planta que produce ES.

La planta fue originalmente diseñada para producir 91 T/D de ES y se pretende aumentar la capacidad para producir 150 T/D. Dentro de las varias alternativas analizadas se pretende efectuar la optimización sobre el diagrama que se muestra a continuación :

C1, C2, C3 : calentadores

E1, E2, E3 : cambiadores

R : reactor catalítico



Bases de diseño

1. Composición de la carga fresca (%)

B —— 0.43

T —— 0.86

E B —— 98.46 ⟵ (materia prima)

P E B —— 0.22

2. Composición de la carga total al reactor (sin vapor)

B —— 0.166

T —— 1.866

E B —— 95.217

E S —— 2.668 ⟵ (producto final)

P E B —— 0.082

Descripción del Flujo

El EB fresco se une con la corriente de recirculación proveniente de otra sección de la planta. La corriente resultante se bombea al sistema de precalentamiento de carga, constituido por los cambiadores E1 y E2 ; antes de entrar a estos cambiadores la carga combinada se mezcla con aproximadamente el 9% del vapor total usado en la reacción. La mezcla, parcialmente vaporizada, se alimenta al cambiador E2 a una temperatura de 316° F , saliendo del mismo a 692° F , completamente vaporizada. Del cambiador E2 para al E1 de donde sale a 1092° F . El calentamiento final de la mezcla vapor-

hidrocarburos se suministra en el calentador C1, previa adición de vapor adicional proveniente del cambiador E3 y del calentador C2, de donde sale a una temperatura de 1150° F.

El 91½ restante del vapor requerido por el proceso se precalienta en el cambiador E3. Este vapor entra a 366° F y sale del cambiador E3 a 748° F. El vapor asi calentado, parte se alimenta al calentador C3, de donde sale a 1300° F y el resto se alimenta al C2 para más tarde mezclarse con la corriente de vapor-hidrocarburos-antes de-entrar al calentador C1.. Por otro lado, el vapor que sale del calentador C3 se divide en dos corrientes, a saber: una parte sirve para dar la temperatura final y la relación (vapor/hidrocarburos) a la entrada del reactor R, mientras que la otra se usa para elevar la temperatura de los gases de reacción entre los hechos catalíticos del reactor.

La mezcla vapor - hidrocarburos que sale del reactor R a 1119° F se aprovecha para precalentar los hidrocarburos y el vapor que entran al proceso.

(Nota: La mayoria de los datos dados con anterioridad son resultado de la optimización que se describe más adelante).

Datos

Carga fresca : cantidad máxima disponible

composición y temperatura

Recirculación : Cantidad y composición y temperatura

Calentador C3 : temperatura de salida

Vapor : temperatura de entrada

EB en producto a separación : cantidad (140 T/D)

Se requiere calcular lo siguiente :

1.. Calcular el volúmen de catalizador en el lecho L 1

2.. Calcular el volúmen de catalizador en el lecho L 2

3.. La cantidad (%) del flujo de salida del reactor R que pasa por los cambiadores E 1 y E 3

4.. La cantidad total de vapor

5.. La cantidad de vapor que se inyecta al cambiador E 2

6.. La cantidad de vapor que pasa por el calentador C 2 para que la mezcla a la salida del calentador C 1 sea de 1150° F

7.. La relación (vapor/hidrocarburos) a la entrada del lecho L 1 del reactor R y a la entrada del lecho L 2 del mismo reactor.

El objetivo es el siguiente

1. Minimizar la cantidad total de catalizador en el reactor R (lechos L1 y L2)

2. Que la producción sea lo más cercana a 140 T/D

3. Que el consumo de carga fresca sea la menor posible (G)

4. Que el producto a separación esté lo más frio posible (T).

Con lo anterior se puede plantear la siguiente función objetivo :

$$F = K_1 (L_1 + L_2) + K_2 (\text{prod} - 150)^2 + K_3 (G) + K_4 (T)$$

donde K_1, K_2, K_3 y K_4 son constantes que se usan para "condicionar adecuadamente" la función objetivo.

Restricciones

Debido a condiciones de operación de los equipos, se deben tomar en cuenta las siguientes restricciones

$$1000°\,F \leq \text{Temp. entrada al reactor} \leq 1280°\,F$$

$$\text{Temp. salida del primer lecho (L1)} \leq \text{Temp. entrada al segundo lecho (L2)} \leq 1250°\,F$$

$$1.0 \leq \text{Relación (Vap./hidrocarburos) entrada al primer lecho} \leq 3.0.$$

$$\text{Relación (Vap./hidrocarburos) entrada al } 1^{er} \text{ lecho} \leq \text{Relación (Vap./hidrocarburos) entrada al } 2^{a} \text{ lecho} \leq 3.0$$

Modelo de los equipos

Los equipos que es necesario simular $^{y}/_{o}$ dimensimar son los siguientes :

1. Calentadores (C1, C2, C3)

2. Cambiadores de calor

   a) Gas - Gas (E1 y E3)

   b) Gas - Liquido con evaporación (E2)

3. Reactor químico catalítico (R)

De los equipos mencionados sólo se describirá el modelo usado para el reactor catalítico, por ser este el equipo más importante de la planta. Para el resto de los equipos los modelos matemáticos son bastante convencionales (Ver., p.ej., D.D. Kern (Process Heat Transfer", Mc Graw-Hill Book Co., New York).

Cinética del sistema reaccionante

Se considera que en los lechos catalíticos L1 y L2 se llevan a cabo las siguientes reacciones. (A : vapor de agua).

1) $EB \rightleftharpoons ES + H$ (catalítica)

2) $EB \longrightarrow B + E$ (catalítica)

3) $H + EB \longrightarrow T + M$ (catalítica)

4) $M + A \longrightarrow CO + H$ (catalítica)

5) $CO + A \longrightarrow CO_2 + H$ (catalítica)

6) $E \longrightarrow AC + H$ (descomposición en fase vapor)

Las expresiones para la velocidad de reacción de cada una de las reacciones anteriores, son las siguientes

$$V_1 = \left(P_{EB} - \frac{P_{ES}\ P_H}{K_P} \right) \exp \left(- \frac{5715}{T} - 6.16 \right)$$

$$V_2 = \exp \left(- \frac{25600}{T} + 12.8 \right) P_{EB}$$

$$V_3 = \exp \left(- \frac{11000}{T} - 1.8 \right) P_{EB}\ P_H$$

$$V_4 = \exp \left(- \frac{7900}{T} - 3.36 \right) P_M$$

$$V_5 = {}^{P} \exp \left(- \frac{8850}{T} + 3.8 \right) P_{co}\ P_A$$

$$V_6 = 2.5 * 10'' \exp \left(- \frac{38000}{T} \right) \cdot P_E/_T$$

donde

$$K_P = T^{0.549} \exp \left(- \frac{14516}{T} + 11.41 \right)$$

P : Presión en atmósferas

T : Temperatura

P_i : Presión parcial a cada componente

El calor liberado por cada una de las reacciones se tomó igual a

$$\Delta H_1 = 28,843 + 1.09\ T$$

$$\Delta H_2 = 25,992 - 1.09\ T$$

$$\Delta H_3 = 12,702 - 3.15\ T$$

$$\Delta H_4 = 50,046 + 3.96\ T$$

$$\Delta H_5 = 10,802 + 2.5\ T$$

$$\Delta H_6 = 38,278 + 11.45\ T$$

Con la anterior información, es posible establecer las relaciones que describen la variación, con la longitud, de cada uno de los componentes así como de la presión y temperatura, las cuales tendrán la forma general

$$\frac{d\,n_i}{d\,Z} = f_i\ (n_1, n_2, \ldots, n_{NC}, T, P) \qquad i = 1,2, \ldots \text{ de componentes.}$$

$$\frac{d\,T}{d\,Z} = g\ (n_1, n_2, \ldots, n_{NC}, T, P)$$

$$\frac{d\,P}{d\,Z} = h\ (n_1, n_2, \ldots, n_{NC}, T, P)$$

con sus condiciones iniciales asociadas ($Z = 0$)

Hay que tomar en cuenta que el volúmen de catalizador en cada uno de los lechos es una incognita, por lo que el límite superior de integración en cada lecho ($Z = Z_1$ y $Z = Z_2$) son variables, además de que entre ambos lechos existe una sección de mezclado con vapor.

Resultados

Los resultados que produjeron un valor mínimo de la función objetivo fueron los siguientes :

Temperatura de entrada a la primera cama	1243° F
Temperatura de entrada a la segunda cama	1145° F
Longitud de la primera cama	211.4 cm
Longitud de la segunda cama	157.1 cm
(Vapor/hidrocarburos) primera cama	2,738 lb/lb
Gasto de hidrocarburos en la primera cama	249.31 lbmol/h
(Vapor/hidrocarburos) segunda cama	2,923 lb/lb
ES a purificación	150.4 T / D
Conversión total	65.06 %
Selectividad total	81.84 %
Fracción de la salida del reactor a E1 y E3	64 %
Temperatura del producto a separación	562° F
Fracción del vapor total al cambiador E 2	8.3 %
Fracción del vapor total al calentador C 3	0 %

Comentarios Finales

El problema anteriormente descrito resulta ser, desde el punto de vista de minimización de funciones, uno de los más complejos ya que una sola evaluación de la función objetivo requiere de los siguientes cálculos

a) Solución de sistemas de ecuaciones diferenciales ordinarias no lineales

b) Cálculos iterativos en los cambiadores de calor ya que estos existen de antemano y los flujos y temperaturas de operación deben ajustarse al diseño mecánico que tienen.

c) Existe dentro del proceso un paso de recuperación de energía (cambiadores de calor), lo cual también genera que se hagan cálculos iterativos dentro de cada evaluación de la función objetiva.

En opinión del autor de este ejemplo, el modificar las condiciones de operación de una planta para ajustarlas a nuevos requerimientos es, con mucho, bastante más complejo que el diseño de una nueva planta ya que en este último caso se tienen muchos grados de libertad.

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

OPTIMACION DE SISTEMAS FISICOS

ESTABILIZACION DEL METODO DE NEWTON PARA LA SOLUCION DE SISTEMAS DE ECUACIONES NO LINEALES

A V Levy *
A C Segura **

* Investigador del IIMAS-UNAM
** Investigador INEN

Marzo, 1981

Resumen.

Un nuevo método general para resolver sistemas de ecuaciones no lineales de una manera eficiente, se ha obtenido. Este método tiene la característica de resolver aquellos problemas en los cuales la singularidad de la matriz jacobiana se presenta: En estos problemas los métodos de Newton y sus variantes dejan de funcionar. El método está basado en la introducción de una nueva función, llamada función de tunelización, cuando se detecta la existencia de una singularidad. Esta función tiene la ventaja de preservar las soluciones de la función original y de eliminar la singularidad de la matriz jacobiana.

La base teórica del presente método está intimamente relacionado, por una parte, con la existencia de puntos singulares. Aunque todavía no existe una explicación teórica definitiva, es evidente de los resultados experimentales que la existencia de una o más singularidades no imposibilitan que el método tenga una alta probabilidad de convergencia y en algunos casos puede ser globalmente convergente. Por otra parte, en la ausencia de puntos singulares, resulta plausible esperar no solamente que el algoritmo sea globalmente convergente, sino que se vuelva idénticamente al Método de Restauración Modificado, Ref. [1]. Esta conjetura está basada en el comportamiento del método. Como un resultado particular de este método se obtiene el Método de Newton Estabilizado, de donde los resultados obtenidos para el primer método serán también válidos para el segundo método.

CONTENIDO.

1. Introducción.

En general los métodos de Newton, ref. [1], presentan malas propiedades de convergencia, puesto que divergen cuando la matriz jacobiana se vuelve singular en algún punto nominal. A tal punto lo llamaremos punto singular o simplemente singularidad y lo denotaremos por x^*. En el presente trabajo se va a desarrollar un nuevo método para evitar la singularidad de la matriz jacobiana, cuando esta se presenta, lográndose la estabilización del método de Newton clásico.

Para eliminar una singularidad cuando se detecta su existencia, se transforma el problema original a otro problema equivalente, de modo que esta transformación preserve la solución del problema original. A esta transformación la llamaremos Función de Tunelización y se denotará por $T(x,k)$ donde k es un número real que lo utilizaremos para eliminar la singularidad x^*. La construcción de la función de Tunelización es como sigue: Supongamos que se desea resolver un sistema de ecuaciones no lineales de la forma

$$\phi(x) = 0 \qquad (1)$$

donde ϕ es una función vectorial de dimensión q y x un vector de dimensión n con $q \leq n$. Supongamos que la ec. (1) tiene al menos una solución. Supongamos también que existe cuando menos un punto

x^*en el cual $\phi_x(x^*)= 0$ y $\phi(x^*) \neq 0$.

Al utilizar el Algoritmo de Restauración Modificado, ref. [1], y el valor nominal x está cerca de x^*, entonces cuando $x \to x^*$, $\phi_x(x^*) \to 0$ produciéndose para $\alpha=1$, α el tamaño del paso, desplazamientos de gran magnitud y en general para satisfacer la propiedad de descenso del índice de comportamiento se necesitará utilizar α sumamente pequeña dando como resultado un avance muy lento del algoritmo y en el caso extremo cuando se llega a tener $\phi_x(x^*)=0$ ya no se logra avance alguno por no existir la inversa de $\phi_x(x^*)$. Considérese ahora un punto $x=x^*+\delta$ muy cercano a x^*, donde δ es un vector muy pequeño de dirección aleatoria. La función de tunelización se define como

$$T(x,k) = \frac{\phi(x)}{[(x-x^*)^T(x-x^*)]^k} \qquad (2)$$

El escalar $R= [(x-x^*)^T(x-x^*)]$ es el factor que nos ayudará a eliminar la singularidad x^*para un valor de k suficientemente grande. Es decir, si $\phi_x(x^*)= 0$ entonces

$$T_x(x,k) = \frac{\phi_x(x)}{R^k} - \frac{2k}{R^{k+1}}[(x-x^*)\phi^T(x)] \qquad (3)$$

$$\text{---} - \frac{2k}{R^{k+1}}[(x-x^*)\phi^T(x)] \neq 0$$

Por lo tanto un nuevo desplazamiento puede obtenerse.

Como resultado se obtiene un Método de Restauración Estabilizado (MRE) cuya propiedad principal es su convergencia cuadrática cuando el punto nominal se elige muy cerca de la solución, permaneciendo en este caso k=0 constante, así como tener una mayor probabilidad de convergencia a la solución, que el método de Restauración sin estabilización. Cuando q=n éste método se transforma en el Método de Cuasilinealización Estabilizado (MCE), donde los resultados obtenidos para el MRE serán válidos también para este Método.

2. Formulación del Problema.

Supóngase que se desea resolver un sistema de ecuaciones no lineales descrito por la ecuación

$$\phi(x) = 0 \qquad (4)$$

donde ϕ es una función vectorial de dimensión q y x un vector de dimensión n, con $q \leq n$. Supongamos que la primera derivada de $\phi(x)$ con respecto a x existe y es continua. Supóngase también que la ecuación tiene una solución.

Indice de Comportamiento. Como la ec. (4) es no lineal, está uno obligado a usar métodos aproximados para resolverla. En particular se usará cuasilinealización. Por lo tanto, es conveniente introducir la función escalar P(x) definida como

$$P(x) = \phi^T(x)\ \phi(x) \qquad (5)$$

llamada índice de comportamiento. La función P(x) mide el error cometido en las ecuaciones al usar métodos aproximados. Para un valor dado de x se tiene

$$P(x) = 0 \quad \text{si} \quad x = x^* \qquad (6)$$

$$P(x) > 0 \quad \text{si} \quad x \neq x^* \qquad (7)$$

donde x^* denota la solución exacta. Si se usa cuasilinealización,

se deben de obtener valor de x tales que

$$P(x) \leq \varepsilon \qquad (8)$$

donde ε es un número pequeño seleccionado de antemano y de acuerdo a la exactitud requerida en la solución.

3. Detección de la Singularidad.

Durante el desarrollo del Método de Restauración Modificado (MRM) observamos que la dificultad principal de este método es la presencia de la singularidad de la matriz jacobiana en determinados problemas. En tales circunstancias al aplicar el MRM a tales problemas, el algoritmo diverge, porque al pedir que la propiedad de descenso se cumpla se necesitará tomar un valor de α sumamente pequeño sin lograr tal propósito. Por lo tanto, teóricamente hemos proporcionado una técnica para detectar la existencia de una singularidad. Desafortunadamente, en la práctica, este proceso es demasiado caro por requerir mucho tiempo de cómputo para llegar exactamente a la singularidad. Sin embargo, podemos dar una técnica de detección bastante aceptable que nos indique la existencia de una singularidad, sin llegar exactamente a ella. Esta técnica consiste en fijar un número máximo de biseccíones en α para satisfacer la propiedad de descenso del índice de comportamiento. Es decir, si Bisec denota el número de biseccíones hechas sobre α y B_{max} el número máximo de biseccíones permitidas sobre α; entonces

Entonces existe una singularidad si Bisec > B_{max} (9)

Entonces no existe una singularidad si Bisec ≤ B_{max} (10)

4. Eliminación de la Singularidad.

Una vez detectada la existencia de una singularidad x^* el camino a seguir es tratar de eliminarla. Para ello reemplacemos a la función original por una función equivalente de tal manera que ésta función tenga la misma solución que la función original.

Función de Tunelización. Para eliminar la singularidad se propone la siguiente función.

$$T(x,k) = \frac{\phi(x)}{[(x-x^*)^T(x-x^*)]^k} \qquad (11)$$

llamada Función de Tunelización. La función $T(x,k)$ es una función vectorial no lineal de dimensión q, $x = x^b + \delta$ un punto cercano a x^* de dimensión n; δ un vector de dirección aleatoria y k un parámetro cuyo valor será determinado para eliminar la singularidad.

Selección del parámetro k. Supongamos que $\phi_x(x^*) = 0$. Derivando la ec. (11) con respecto a x, obtenemos

$$T_x(x,k) = \frac{1}{R^k}[\phi_x(x)] - \frac{2k}{R^{k+1}}[(x-x^*)\phi^T(x)] \qquad (12)$$

donde $T_x(x,k)$ es una matriz de dimensión n x q.

Si k=0, entonces las ecs. (11) y (12) se tranforman en

$$\begin{aligned} T(x,k) &= \phi(x) \\ T_x(x,k) &= \phi_x(x) \end{aligned} \qquad (13)$$

que coinciden con la formulación del NRM. Como $\phi_x(x^\circ) = 0$, se tiene $T_x(x,k) = 0$, produciéndose la divergencia del ARM para k=0.

Para poder calcular un nuevo desplazamiento es necesario que $T_x(x,k) \neq 0$. Ahora, incrementando k en incrementos $\Delta k > 0$, tal que, $k=k + \Delta k$, se tiene

$$T_x(x,k) = \frac{1}{0(\delta^{2k})}[\phi_x(x^\circ)] - \frac{2k}{0(\delta^{2k+2})}[0(\delta)\,\phi^T(x)] \qquad (14)$$

Como $\phi_i(x) \neq 0$, $i=1,2,\ldots,q$, obtenemos

$$T_x(x,k) = -\frac{2k}{0(\delta^{2k+2})}[0(\delta)\,\phi^T(x)] \neq 0 \qquad (15)$$

que era lo que se pedía para poder calcular un nuevo desplazamiento, para un valor de k suficientemente grande.

En el análisis anterior hemos supuesto que

$$||x-x^\circ|| < 1 \qquad (16)$$

de modo que, para valores de k muy grande

$$\frac{1}{|(x-x^\circ)^T(x-x^\circ)|^k} \neq 0 \qquad (17)$$

Por lo tanto, en la vecindad de un punto x°, se tiene

$$|T_x(x,k)| \longrightarrow -\frac{2k}{R^{k+1}}|(x-x^\circ)\phi^T(x)| \neq 0 \qquad (18)$$

para k suficientemente grande.

Si la ec. (16) no se cumple entonces la singularidad de x° se sustituye por el último punto x encontrado, generándose a continuación el vector $x=x^\circ+\delta$.

El procedimiento anterior para determinar el valor de k puede resumirse como sigue: Supóngase que para k=0 hemos descubierto una singularidad por el procedimiento de la sec. (3). Entonces

(i) — Calcular $T(x,k)$ y $T_x(x;k)$.

(ii) Si la ec. (16) se cumple pasar al paso (iv). En caso contrario pasar a (iii).

(iii) Efectuar la asignación

$x^\circ = x$

$x = x^\circ + \delta$

$k = 0$

regresar a (i)

(iv) Incrementar k, en incrementos $\Delta k > 0$. Regresar a (i).

5. Algoritmo de Restauración Estabilizado.

A continuación vamos a prestar el desarrollo del nuevo algoritmo para obtener el siguiente desplazamiento, después de haber detectado y eliminado una singularidad x^*. La característica de este algoritmo es una probabilidad de convergencia más grande que la del método de Restauración Modificado.

Indice de Comportamiento. Como la ec. (11) es también no lineal, está uno obligado a usar métodos aproximados para resolverla. Por lo tanto, es importante introducir el índice de comportamiento $Q(x, k)$, definido por

$$Q(x, k) = T^T(x, k)\ T(x, k) \tag{19}$$

que mide el error en las ecuaciones al usar métodos aproximados. Así dado cualquier punto nominal x se tiene

$$Q(x,k) = 0 \quad si \quad x=x^* \tag{20}$$

$$Q(x,k) > 0 \quad si \quad x\neq x^* \tag{21}$$

donde x^* es la solución exacta. Si se usa cuasilinealización se deben de obtener valores de x tales que

$$Q(x,k) \leq \varepsilon \tag{22}$$

donde ε es un número muy pequeño seleccionado de acuerdo a la exactitud requerida en la solución.

Como el problema original es resolver la ec. (4) y T(x,k) es una ecuación derivada de ésta, resulta natural considerar como criterio de convergencia del algoritmo la ec. (8), puesto que para $k > 0$ se tiene en general $P(x) < Q(x,k)$, por lo que, el criterio de convergencia (22) es reemplazado por la ec.(8).

Sea $x = x^{\circ} + \delta$ un punto nominal dado. Considérese el desplazamiento Δx que nos lleva del punto nominal x a un punto variado $\tilde{x}$, tal que

$$\tilde{x} = x + \Delta x \tag{23}$$

Si se usa cuasilinealización, entonces la ec. (11) es aproximada por

$$\delta T(x,k) = -\alpha\, T(x,k), \qquad 0 < \alpha \leq 1 \tag{24}$$

donde

$$\delta T_x(x,k) = T_x^T(x,k)\, \Delta x \tag{25}$$

denota la primera variación de $T(x,k)$. $T_x(x,k)$ representa una matriz de dimensión $n \times q$, cuya j-ésima columna es el gradiente de la función $T_j(x,k)$, $j=1,2,\ldots,q$, con respecto al vector x.

Sustituyendo la ec. (25) en la ec. (24) obtenemos

$$T_x^T(x,k)\, \Delta x = -\alpha\, T(x,k), \quad 0 < \alpha \leq 1 \tag{26}$$

la ec. (26) representa un sistema lineal algebraico de q ecuaciones con n incógnitas. Si hacemos la hipótesis $q < n$, entonces la ec.(26) tiene la infinidad de soluciones Ref. [ ]. Sin embargo, se puede obtener una solución única si se dá la siguiente condición: "Una solución única de la ec. (26) se puede obtener si se busca un desplazamiento de longitud mínima en el sentido del principio de mínimos cuadrados"

Por consiguiente se desea resolver el siguiente problema.

$$\text{Min} \quad w = \frac{1}{2} \Delta x^T \Delta x \tag{27}$$

s.a.

$$T_x^T(x,k)\ \Delta x + \alpha\ T(x,k) = 0 \tag{28}$$

Por los métodos de la teoría de máximos y mínimos, se sabe que el problema anterior puede ser reformulado como el minimizar la función aumentada,

$$F(\Delta x,\lambda) = \frac{1}{2} \Delta x^T \Delta x + \lambda^T \{T_x^T(x,k)\Delta x + \alpha T(x,k)\} \tag{29}$$

donde $F(\Delta x,\lambda)$ es conocida como la función Lagrangiana y el vector λ de dimensión q como el multiplicador de Lagrange. Si

$$F_{\Delta x}(\Delta x,\lambda) = \Delta x + T_x(x,k)\lambda \tag{30}$$

denota el gradiente de la función aumentado $F(\Delta x,x)$ con respecto

al vector Δx, el desplazamiento óptimo debe satisfacer

$$F_{\Delta x}(\Delta x, \lambda) = 0 \qquad (31)$$

Esto produce la relación

$$\Delta x = - T_x(x,k)\,\lambda \qquad (32)$$

sustituyendo la ec. (32) en la ec. (26) se obtiene

$$A(x,k)\,\lambda = \alpha\, T(x,k) \qquad (33)$$

donde $A(x,k)$ es una matriz de dimensión $q \times q$ de la forma

$$A(x,k) = T_x^T(x;k)\; T_x(x,k) \qquad (34)$$

Para un valor dado de α la ec. (33) representa un sistema lineal de q ecuaciones con q incógnitas, y puede resolverse por eliminación gausiana para obtener el valor de λ.

Cambio de Coordenadas. Para simplificar el problema anterior introducimos la variable auxiliar

$$Z = \frac{\lambda}{\alpha} \qquad (35)$$

transformándose la ec. (33) en

$$A(x,k)\; Z = T(x,k) \qquad (36)$$

que es equivalente a un sistema lineal algebraico en la incógnita Z.

Para obtener el valor de Z resolveremos la ec. (36) por eliminación gausiana, e inmediatamente después introducimos otra variable auxiliar $\underline{Y}$ definida por

$$Y = - T_x(x,k)\ Z \tag{37}$$

donde Y es un vector de dirección. El valor del desplazamiento Δx puede obtenerse de la ecuación

$$\Delta x = - \alpha\ \underline{Y} \tag{38}$$

El vector $\tilde{x}$ es calculado de la ec.(23). Si el punto $\tilde{x}$ satisface la ec. (8) entonces $\tilde{x}$ es la solución buscada y el algoritmo se termina. En caso contrario el punto $\tilde{x}$ se toma como punto nominal para la siguiente iteración. Como resultado obtenemos un Algoritmo de Restauración Estabilizado (ARE) y es usado iterativamente hasta lograr la convergencia a la solución buscada.

Si q=n, de la ec. (36) obtenemos

$$Z = A^{-1}(x,k)\ T(x,k) \tag{39}$$

$$= T_x^{-1}(x,k)\ [T_x^T(x,k)]^{-1}\ T(x,k)$$

Sustituyendo la ec. (39) en la ec. (37) obtenemos

$$Y = - [\ T_x^T\ (x,k)]^{-1}\ T(x,k) \qquad (40)$$

Multiplicando por la izquierda la ec. (40) por $T_x^T(x,k)$, obtenemos

$$T_x^T(x,k)\ \underline{Y} = - T(x,k) \qquad (41)$$

La ecuación (41) es equivalente a un sistema lineal de n ecuaciones con n incógnitas y puede resolverse por eliminación gaussiana para obtener el valor de $\underline{Y}$. Una vez conocido el vector $\underline{Y}$ calcular Δx y $\tilde{x}$ de las ecuaciones (38) y (23) respectivamente. Si $\tilde{x}$ satisface la ec. (8), entonces $\tilde{x}$ es la solución buscada y el algoritmo si termina. En caso contrario, se toma a $\tilde{x}$ como punto nominal y el proceso vuelve a repetirse. Como resultado obtenemos el Algoritmo de Cuasilinealización Estabilizado (ACE), o algoritmo de Newton Estabilizado.

Propiedad de Descenso del Indice de Comportamiento. Para impedir que el índice de comportamiento $Q(x,k)$ aumente cuando pasamos del punto nominal x al punto variado $\tilde{x}$, requerimos que su primera variación sea negativo. La primera variación de $Q(x,k)$ está dado por

$$\delta Q(x,k) = 2\ T^T(x,k)\ \delta T(x,k) \qquad (42)$$

De la ec. (24), se tiene

$$\delta Q(x,k) = -2\alpha Q(x,k) \qquad (43)$$

Ahora, como $Q(x,k) > 0$, ya que x es un punto que no satisface la ec. (11), y para $\alpha > 0$, tenemos.

$$\delta Q(x,k) < 0 \qquad (44)$$

Por lo tanto, si α es muy pequeño el descenso del índice de comportamiento está garantizado, es decir

$$\bar{Q}(\bar{x},k) < Q(x,k) \qquad (45)$$

Las ecs. (44) ÷ (45) constituyen lo que llamamos la propiedad de descenso del índice de comportamiento. Para determinar el valor óptimo de α para el cual la propiedad de descenso se cumpla, se efectuará un proceso de bisección sobre α , ref. [1]

6. Resumen de los algoritmos.

Las diferentes etapas del desarrollo de los algoritmos; ACE y ARE pueden resumirse de la siguiente manera:

a) Algoritmo de Cuasilinealización Estabilizado.

1. Dar como punto nominal a x°.
2. Generar el punto $x=x^\circ+\delta$. Asignar $k=0$.
3. Calcular $\phi(x)$, $\phi_x(x)$ y $P(x)$.
4. Si $P(x)$ satisface la ec. (8), entonces el algoritmo se termina. En caso contrario pasar al siguiente paso.
5. Calcular $T(x,k)$, $T_x(x,k)$ y $Q(x,k)$.
6. Calcular Y de la ec. (41). Asignar $\alpha=1$.
7. Calcular Δx y $\tilde{x}$ de las ecs. (38) y (23), respectivamente.
8. Calcular $\phi(\tilde{x})$, $T(\tilde{x},k)$ y $\tilde{Q}(\tilde{x},k)$.
9. Si $\tilde{Q}(\tilde{x},k) < Q(x,k)$ pasar al paso (13). En caso contrario pasar al siguiente paso.
10. Si la ec. (9) se cumple pasar al paso (11). En caso contrario el valor de α se reemplaza por $\alpha/2$ y se regresa a (7).
11. Si la ec. (16) se cumple pasar a (12). En caso contrario efectuar las asignaciones $x^\circ = x$, $x=x^\circ+\delta$ y $k=0$. Regresar a (3).
12. Incrementar k en incrementos $\Delta k > 0$. Regresar a (5).
13. Una vez conocido $\tilde{x}$ la iteración se termina. El punto $\tilde{x}$ se toma como punto nominal para la siguiente iteración. Regresar a (3).

b) Algoritmo de Restauración Estabilizado.

1. Dar como un punto nominal a x°. Generar el punto $x = x^\circ + \delta$.
   Asignar k=0.
2. Calcular $\phi(x)$, $\phi_x(x)$ y $P(x)$.
3. Si $P(x)$ satisface la ec. (8) el algoritmo se termina. En caso contrario pasar al siguiente paso.
4. Calcular $T(x,k)$, $T_x(x,k)$ y $Q(x,k)$.
5. Calcular $\hat{Z}$ y Y de las ecs. (36) y (37) respectivamente. Asignar $\alpha=1$.
6. Calcular Δx y $\tilde{x}$ de las ecuaciones (38) y (23) respectivamente.
7. Calcular $\phi(\tilde{x})$, $T(\tilde{x},k)$ y $\tilde{Q}(\tilde{x},k)$.
8. Si $\tilde{Q}(\tilde{x},k) < Q(x,k)$ pasar al paso (12). En caso contrario pasar al siguiente paso.
9. Si la ec. (9) se cumple pasar al paso (10). En caso contrario el valor de α es reemplazado por $\alpha/2$! Regresar a (6).
10. Si la ec. (16) se cumple pasar al paso (11). En caso contrario efectuar las asignaciones $x^\circ = x$, $x = x^\circ + \delta$ y k=0. Regresar a (2).
11. Incrementar k en incrementos $\Delta k > 0$. Regresar a (4).
12. Con $\tilde{x}$ conocido la iteración se termina. Tomar a $\tilde{x}$ como punto nominal para la siguiente iteración, regresando a (2).

8. Conclusiones.

Un método general para resolver sistemas de ecuaciones no lineales de la forma $\phi(x) = 0$, donde $\phi(x)$ es una función vectorial de dimensión q y x un vector de dimensión n con $q \leq n$, se ha desarrollado. Este método tiene la característica de resolver aquellos problemas en los cuales la matriz $\phi_x(x^*)=0$ se presenta, identificando a x^* como un punto singular. La eliminación del punto singular x^* se logra al introducir la función de tunelización $T(x,k)$ para un valor de k suficientemente grande. El método está basado en la consideración de los indices de comportamiento $P(x)=\phi^T(x)\ \phi(x)$ y $Q(x,k) = T^T(x,k)\ T(x,k)$, donde el primero es usado como criterio de convergencia y el segundo como una guía durante el desarrollo del método, ya que en general, para $k > 0$ se tiene $Q(x,k) > P(x)$.

Un algoritmo de Restauración Estabilizado ha sido generado al considerar la existencia y eliminación de puntos singulares y al mismo tiempo requerir que la primera variación del índice de comportamiento $Q(x,k)$ sea negativo.

Si k=0 el ARE se transforma identicamente en el Algoritmo de Restauración Modificado (ARM), el cual no tiene la propiedad de detectar y eliminar puntos singulares. Esto significa que si el ARE es usado con k=0 entonces el algoritmo no puede (en general) converger a la solución.

La propiedad fundamental del ARE, la capacidad de poder detectar l existencia de puntos singulares y la capacidad de poder eliminarlos,

hace que el algoritmo sea más confiable puesto que, en algunos casos puede volverse globalmente convergente.

Finalmente, cuando q=n el ARE se transforma en el Algoritmo la Cuasilinealización Estabilizado y los resultados obtenidos para el primero son también válidos para el segundo.

8. Condiciones Experimentales.

Con el objeto de comparar las ventajas logradas con los algoritmos ACE y ARE sobre los algoritmos ACO, ACM y ARO y ARM, respectivemente, se resolvieron varios ejemplos numéricos usando una computadora B6700 con aritmética de doble precisión. Todos los algoritmos fueron programados en Fortran IV.

Valores nominales. Para todos los algoritmos los valores nominales fueron elegidos como

$$x_{ij} = x_i^* + \beta\, \delta_{ij} \quad i=1, 2, \ldots n; \; j=1,2,\ldots,2n \qquad (46)$$

usando los siguientes valores de β y de δ_{ij}

$$\beta = 0, \pm 1, \pm 2, \ldots, \pm 50 \qquad (47)$$

$$\delta_{ij} = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases} \qquad (48)$$

En la ec. (46), x^* representa la solución exacta, β es la distancia del punto nominal x a la solución exacta x^* y δ_{ij} es la dirección tomada sobre los ejes coordenados.

Condición de Paro. El criterio de convergencia para obtener la solución deseada, se escogió para todos los algoritmos como

$$P(x) \leq 10^{-20} \qquad (49)$$

y los criterios de no convergencia como

$$N \geq 100 \qquad (50.1)$$

$$N_b \geq 20 \qquad (50.2)$$

$$K_m \geq 10 \qquad (50.3)$$

donde N es el número máximo de iteraciones del algoritmo para lograr la convergencia a la solución deseada, N_b es el número máximo de bisecciones permitidas en el tamaño del paso α para satisfacer la propiedad de descenso de cada algoritmo y K_m el valor máximo permitido del parámetro k de la ec. (11).

La condición (50.1) implica la convergencia muy lenta del algoritmo, la condición (50.2) indica un valor extremadamente pequeño del desplazamiento y por consecuencia la convergentia muy lenta, y la condición (50.3) indica que los algoritmos ACE y ARE no pudieron cancelar la singularidad.

El valor de Δk es 0.1 y el valor de B_{max} en las ecs. (9-10) es 5 y 15 para el ACE y el ARE, respectivamente. El valor de δ es 10^{-3}.

Porcentaje de Exito. Puesto que cada problema fue resuelto varias veces (100 en total) comenzando con los valores nominales dados por las ecuaciones (46)- (48), sean N_r el número total de corridas efectuadas para un cierto problema dado usando un algoritmo dado, y N_s el número total de corridas exitosas. El porcentaje de éxito está definido por

$$p = \frac{N_s}{N_r}$$

de donde $0 \leq p \leq 1$ y m(entras más grande sea el valor de p más poderoso es el algoritmo.

Radio de Convergencia. Para hacer una comparación de la eficiencia de cada algoritmo, definimos el radio de convergencia de la siguiente manera: Dado un punto nominal x la solución conocida x*, medimos la distancia de x a x* como $R=||x^* - x^*|| \epsilon \beta \delta_{ij}$. Si para todos los nominales se tiene p=1, decimos entonces que R(R=β) es el radio de convergencia de un algoritmo dado para un problema dado.

Tiempo de Computo Empleado. Si T_i representa el tiempo en segundos de CPU emplado en la i-ésima corrida exitosa, para resolver un problema dado usando un algoritmo dado, definimos el tiempo promedio en segundos de CPU por corrida exitosa como

$$T_{av} = \frac{1}{N_s} \sum_{i=1}^{N_s} T_i$$

9. Ejemplos numéricos.

En esta sección se describen 13 ejemplos números. Por simplicidad se usará notación escalar.

Ejemplo 1.1 Considérese le ecuación no lineal

$$x - \sin 3x + 1$$

Con solución conocida x= -1.04

Ejemplo 1.2 Considérese la ecuación no lineal

$$x - \text{sen}(2x) = 0$$

con soluciones conocidas de $x=0; \pm 0.95$

Ejemplo 1.3 Considérese la ecuación no lineal

$$\text{sen}\,(x) - x + 2 = 0$$

con solución conocida $x = 2.56$

Ejemplo 1.4 Considérese el sistema no lineal

$$e^{\text{sen}\,(x^3)} + x - 1 = 0$$

$$2x^2 + 3y^2 - 4xy + 8(y-x) - 1 = 0$$

con solución conocida $(x,y) = (0,0.12)$. Otras soluciones son por ejemplo $(x,y) = (0,-2.79)$, $(-1.65, -2.23)$.

Ejemplo 1.5 Considérese el sistema no lineal [25]

$$x^2 + x - y^2 - 1 = 0$$

$$y - \text{sen}\,(x^2) = 0$$

con solución conocida $(x,y) = (0.73, 0.5)$. Admite también la

solución $(x,y) = (-1.67, 0.35)$.

Ejemplo 1.6 Considérese el sistema no lineal

$$0.05 \operatorname{sen}(4\pi y) - x - 2y + 1 = 0$$

$$y - 0.5 \operatorname{sen}(2\pi x) = 0$$

con solución $(x,y) = (1;0)$. Admite también la solución $(x,y) =$ $(1.6, -0.29)$, $(0.4, 0.29)$, $(1.85, -0.4)$, $(0.15; 0.4)$.

Ejemplo 1.7 Considérese el sistema no lineal

$$2 \operatorname{sen}(\pi x) \operatorname{sen}(2\pi z) - y + 1 = 0$$

$$0.1\, y \operatorname{sen}(2\pi z) - 1.5\, x - z + 2.5 = 0$$

$$0.1\, y \operatorname{sen}(2\pi x) - z + 1 = 0$$

con solución conocida $(x,y,z) = (1,1,1)$

Ejemplo 1.8 Considérese el sistema no lineal

$$2 \operatorname{sen}(0.4\pi x) \operatorname{sen}(0.4\pi z) - y = 0$$

$$0.1z \operatorname{sen}(2\pi z) - x - z + 2.5 = 0$$

$$0.1y \operatorname{sen}(2\pi x) - z + 1 = 0$$

con solución conocida $(x,y,z) = (1.4,1.8,1)$.

Ejemplo 1. 9 Considérese el sistema no lineal

$$2\text{sen}(0.4\Pi x)\ \text{sen}\ (0.4\Pi x) - y = 0$$

$$0.1\ y\ \text{sen}(2\Pi z) - x - z + 2.5 = 0$$

$$0.1y + \text{sen}\ (2\Pi x) - z + 1 = 0$$

con solución $(x;y,z) = (1.05,1.85,1.47)$. Admite también las soluciones $(x,y;z) = (1.91,0.87,056)$, $(1.56,1.55,0.79)$.

Ejemplo 1.10 Considérese el sistema no lineal [2]

$$(x-z)^2 + (y-u)^2 + (x+y+z+u)^2 - 16 = 0$$

$$x\text{sen}(0.5\Pi z) + y\text{cos}(0.5\Pi u) - 1 = 0$$

$$x + y^2 + z^3 + u^4 - 4 = 0$$

$$x + 2y + 3z + 4u - 10 = 0$$

con solución conocida $(x,y,z,u) = (1,1,1,1)$.

Ejemplo 1. 11 Considérese el sistema no lineal [2]

$$(x-y)^2 + (y-z)^2 + (2z-u-v)^2 = 0$$

$$x^2 + y^2 + z^2 + u^2 + v^2 - 5 = 0$$

$$(x-1)^2 + (y-2)^2 + v^4 - 2 = 0$$

$$x + 2y^2 + 3z^3 + 4u^4 + 5v^5 - 15 = 0$$

$$x^2 + xyz - u^3 - 1 = 0$$

con solución conocida $(x,y,z,u,v) = (1,1,1,1,1)$.

Ejemplo 1. 1 Considérese la ecuación no lineal

$$e^{\text{sen}(x^3)} + x - (y-1)^2 - 1 = 0$$

con solución conocida $(x,y) = (0,1)$. Además, admite también las soluciones $(x,y) = (0.26, 1.53)$, $(0.26, 0.4683)$

Ejemplo II.2 Considérese la ecuación no lineal

$$e^{\operatorname{sen}(x^3)} + x - (y-1)^2 - (z-2)^2 - 1 = 0$$

con solución $(x,y,z) = (0,1,2)$. Además tiene más soluciones, por ejemplo $(x,y,z) = (0.26,1,53,2)$, $(0.26,1;2.53)$, $(0.26,0.47,2)$.

Numerical Experiments — Stabilized Newton

Fortran IV . B-6700 Double precision

Convergence

$P(x) < 10^{-20}$

Non-convergence

$N \geq 100$, $N_0 \geq 20$, $\lambda \geq 10$ $(\delta\lambda = 0.1$

Each problem solved with 100 nominal values

$$x_0^i = x^{i*} + \beta \delta_{ij} \qquad i = 1, 2, \ldots n, \quad j = 1, 2 \ldots 2n$$

$$\beta = 0, \pm 1, \pm 2, \pm 3, \ldots \pm 50$$

Probability of Success

$$p = \frac{N_s}{100}$$

N_s : N° of succesful runs.

Average Computing Time

$$T_{av} = \frac{1}{N_s} \sum_{i=1}^{N_s} T_i$$

T_i : CPU Time in seconds

Overall Probability of Success

$$\tilde{p} = \frac{1}{N_{ex}^{\circ}} \sum_{i=1}^{N_{ex}^{\circ}} p_i$$

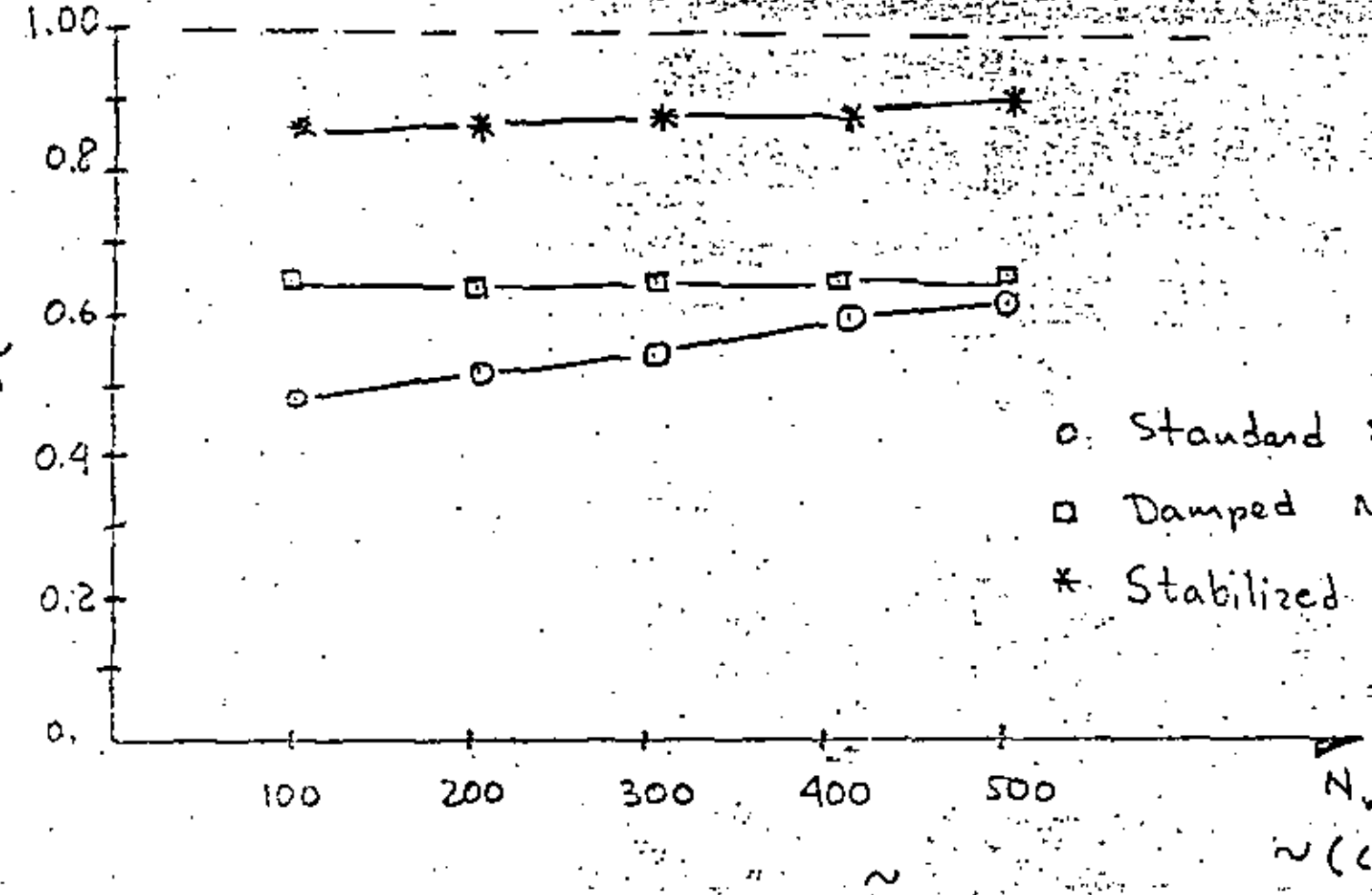
Overall Average Computing Time

$$\tilde{T}_{av} = \frac{1}{N_{ex}^{\circ}} \sum_{i=1}^{N_{ex}^{\circ}} T_i$$

	$\tilde{p}$	$\tilde{T}_{av}$
Standard Newton	0.49	0.441
Damped Newton	0.67	0.370
"Stretched" Newton	0.86	0.993

Ex. No	Newton		Damped Newton		Stabilized Newton	
	p	Tav	p	Tav	p	Tav
1	0.78	1.29	0.72	0.18	1.00	0.48
2	0.67	0.37	0.98	0.10	1.00	0.22
3	0.81	0.47	1.00	0.08	1.00	0.16
4	0.56	0.41	0.53	0.69	0.81	6.60
5	0.42	1.46	0.79	0.27	1.00	0.26
6	0.67	0.41	0.94	0.28	1.00	0.56
7	0.98	0.92	0.84	0.23	1.00	1.08
8	0.44	0.07	0.78	0.33	0.99	0.84
9	0.49	0.24	0.50	0.15	0.96	1.92
10	0.41	3.97	0.29	0.92	0.63	3.80
11	0.76	10.20	0.21	0.96	0.97	7.47

Overall Probability of Success



	$\tilde{p}$
Standard Newton	0.64
Damped Newton	0.68
Stabilized Newton	0.91

Ref: A.V. LEVY & A. SEGURA DUNDEE (1979)

BIBLIOGRAFIA.

1. A. MIELE, S. NAQUI, A.V. LEVY, R.R.IYER. Numerical Solution of Non-linear Equations and Noh-linear Two-Point Boundary Value Problems. Department of Mechanical and Engineering and Materials Science. Rice University, Houston, Texas.

2. A. MIELE, H. Y. HUANG and J.C. HEIDEMAN. Sequential Gradient-Restoration Algorithm for the Minimization of Constrained Function. Ordinary and Conjugate Gradient Versions. JOTA, Vol. 4, No. 4; 1969.

3. K. M. BROWN and J. E. DENNIS. On Newton-Like Iterations Function: General Convergence Theorems and a Specifics Algorithms. Numerische Mathematicke, 12, 1968, Pags. 186-191.

4. J. E. DENNIS. On Newton-Like Methods. Numerische Mathematike, 11, 1968, Pags. 324-330.

5. J. E. DENNIS. On the Kantorovich Hypothesis for Newton Method. SIAM J. on Numerical Analysis, 6, 1969, Pags. 495-507.

6. J. E. DENNIS. On the Newton Method and Nonlinear Symultaneous Displacements. SIAM J. on Numerical Analysis, Serv. B., Vol. 4, No. 1, 1967.

7. L. V. KANTOROVICH and G. P. AKILOV. Functional Analysis in Normed Spaces. N. Y. Pergammon Press, 1964.

8. M. M. VAINVERG. Varational Methods for the Study of Non linear Operator. Holden Day, Inc., San Francisco, 1964.

9. J. TODD. A Sulvery of Numerical Analysis. McGraw-Hill Book Company, New York.

10. J. S. LEE. Quasilinearization and Invariant Imbedding. Academic Press, New York, 1968.

11. D.G.LUENBERGER. Optimization by Vector Spaces Methods. John Wiley, New York, 1969, Pags. 277-281.

12. E. K. BLUM. Numerical Analysis and Computational Theory and Practice. Reading, Mass., Addison-Wesley, 1972, pags. 178-183.

13. B. NOBLE. Applied Linear Algebra. Prentice-Hall, Inc. 1969, pags. 77.

14. G. E. FORSYTHE, and C. B. MOLER. Computer Solutions of Linear Algebraic Systems. Prentice-Hall; Inc. Englewood Cliffs, N. J., 1967.

15. J. M. ORTEGA, and W. C. RHEINBOLDT. Iterative Solution of non Linear Equations in Several Variables. Academic Press, Inc:, New York, 1970.

16. A. M. OSTROWSKI. Solution of Equation and Systems of Equations. Prentice-Hall, Inc. Englewood Cliffs, N. J., 1966.

17. J. F. TRAUB. Iterative Methods for the Solution of Equations. Prentice-Hall, Inc., Englewood Cliffs, N. J., 1964.

18. E. ISAACSON and H. B. KELLER. Analysis of Numerical Methods. John Wiley, New York, 1966.

19. F. J. ZELEZNIK. Quasi-Newton Methods for Nonlinear Equations. J. ACM. Vol. 15, No. 2, 1968.

20. F. FREUDENSTEIN and B. ROTH: Numerical Solution of Systems of Non linear Equations. J. ACM, Vol. 10, No. 4, 1963.

21. K. M. BROWN and W. GEARHART: Deflation Techniques for the Calculation of Further Solution of a Nonlinear Systems. Numerische Mathematiske, 16, 1971.

22. K. M. BROWN. Solution of Simultaneous Non-linear Equations. C. ACM. Vol. 10, No. 11, 1967.

23. F. H. BRANIN. Widely Convergent Method for Find Multiple Solution of Symultaneous Non linear Equations. IBM J. Res. Develop, 1972.

24. R. D. BRENT: On the Davidenko-Branin Method for Solving Symultaneous Non linear Equations. IBM J. Res. Develop. 1972.

25. C. F. GERALD. Applied Numerical Analysis- Addison-Wesley Reading Mass., 1973.

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

OPTIMACION DE SISTEMAS FISICOS

OPTIMACION DE CIRCUITOS ELECTRICOS

DR. MARCO ANTONIO MURRAY-LASSO

MARZO, 1981

REFERENCES

1. Scheerer, W. G.: Optimization with Computers, *Proc. Tutorial Symp. Circuit Design by Computers*, New York Univ., Department of Electrical Engineering, Feb. 1, 1967.
2. Fleischer, P. E.: Optimization Techniques, in F. F. Kuo and J. F. Kaiser (eds.), "System Analysis by Digital Computer," pp. 175-217, John Wiley & Sons, Inc., New York, 1966.
3. Linvill, J. G., and J. F. Gibbons: "Transistors and Active Circuits," McGraw-Hill Book Company, New York, 1961.
4. Llewellyn, F. B.: Some Fundamental Properties of Transmission Systems, *Proc. IRE*, vol. 40, pp. 271-282, March, 1952.
5. Rollet, J. M.: Stability and Power Gain Invariants of Linear Two-Ports, *IRE Trans. Circuit Theory*, pp. 22-32, March, 1962.
6. Raisbeck, G.: Definition of Passive Linear Networks in Terms of Time and Energy, *J. Appl. Phys.*, vol. 25, pp. 1510-1514, Dec., 1954.
7. Owens, J. L.: The Discrete Tantalum Nitride Thin Film Resistor on a Flat Embossed Ceramic Substrate, *Proc. Electron. Components Conf.*, Washington, May 5-7, 1965.
8. Wyndrum, R. W., Jr., and W. Pendergast: A Tantalum Film Gigacycle Amplifier, *Proc. Intern. Solid-State Circuits Conf.*, Philadelphia, pp. 20-21, 1965.
9. Kuo, C. Y., J. S. Fischer, and J. C. King: Thermal Processing of Tantalum Nitride Resistors, *Proc. Electron. Components Conf., Washington*, May 5-7, 1965.
10. Owens, J. L., and N. G. Lesh: Further Developments in Discrete Tantalum Nitride Thin Film Resistors, *ibid.*
11. Wilde, D. J.: "Optimum Seeking Methods," Prentice-Hall, Inc., Englewood Cliffs, N. J., 1964.

4 ANALYSIS OF LINEAR INTEGRATED CIRCUITS BY DIGITAL COMPUTER USING BLACK-BOX TECHNIQUES

M. A. Murray-Lasso

Member, Technical Staff
Bell Telephone Laboratories, Inc.
Whippany, N. J.

4-1 INTRODUCTION

This chapter covers some methods of dealing efficiently wit linear integrated circuits using black-box techniques and describe some computer programs implementing the methods discussed.

One of the drawbacks of the general-purpose analysis program which are presently available (ECAP, NET-1, CIRCUS, PREDICT NASAP, SCEPTRE, etc.) [1-6] is that they can handle only lumped circuit elements. As a result, the models for transistors and in tegrated circuits that the circuit designer is forced to use hav parameters which are difficult to determine and it is a relatively complex matter to achieve good approximations to measured dat over broad frequency ranges. For example, the circuit shown i Fig. 4-1 is a hybrid-pi linear model of a high-frequency transisto [7] including header and overlap diode capacitances which, in spit of its complexity, predicts the frequency behavior of y_{ie} with pre cision up to only a few MHz. For this reason, some designers us different models for each frequency range.

In [8] Carlin emphasizes the fact that one can no longer fa back on the comfortable security of coils, capacitors, resistors

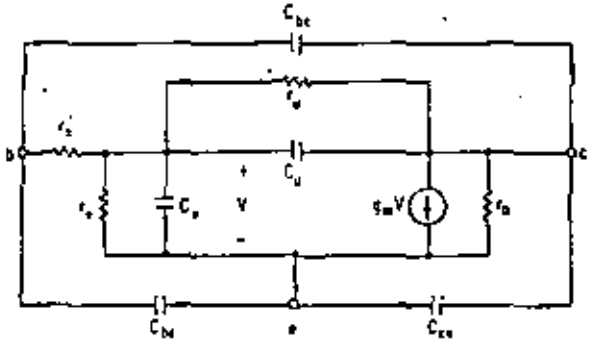


Fig. 4-1 Hybrid-pi model of a transistor with header and overlap diode capacitances.

and transformers in describing the details of construction of complex electrical devices. Independent of the internal construction of a device, if the circuit is linear and time-invariant, frequency response matrices in the complex plane will describe the device accurately. In some cases the frequency response matrix may be obtained experimentally without any knowledge of the internal structure. In other cases integrated circuits may be analyzed and the network may be characterized by a matrix whose entries are functions of frequency. The circuit's internal structure is ignored thereafter. This technique is called the black-box approach. It has the following advantages for integrated circuit analysis

1. It is able to accept experimental as well as analytical data.
2. Complete integrated circuits may be black-box modeled. This is extremely advantageous if the same circuits appear several times in a system.
3. With this approach there is no fundamental difference between lumped, distributed, or ideal networks (such as ideal low-pass filters).
4. Solution of networks by pieces may be done quite simply using this approach. This may become mandatory for analyzing circuits with many nodes on computers with limited memories.

We do not propose to abandon conventional device models since they are quite valuable for giving the designer insight into the interaction of the device with the rest of the circuit. What we propose is that a tool, such as the computer, should be used in ways appropriate to itself. Certain intuitive tools such as circuit

schematics and Bode diagrams are appropriate for the pencil-and-paper designers. A computer, on the other hand, handles polynomials better than logarithmic graphs; it is simpler to fit experimental curves with polynomials than with circuit functions having certain pole-zero constellations. This does not include the designer from keeping in front of him a schematic with conventional models for insight.

The black-box curve modeling approach is most appropriate for handling interconnections involving single-chip linear integrated circuits; the only available points for connection, measurement, and characterization are the external terminals.

The four central ideas in this chapter are

1. In analyzing general linear stationary networks it should be possible to handle them as black boxes.
2. The modeling of a circuit with two or more terminals should be flexible enough to admit not only ratios of complex polynomials, but also general curves in the frequency domain.
3. It should be possible to preanalyze pieces of a circuit and eventually interconnect the pieces.
4. The indefinite matrices [12] are ideal vehicles for characterizing multiterminal black-box circuits which are arbitrarily interconnected.

4-2 FITTING FREQUENCY CURVES WITH STANDARD FUNCTIONS

For the analysis of interconnected black boxes, three methods are possible

1. Analytical expressions which give the terminal characteristics of the boxes.
2. Standard functions fitted to discrete data which were either measured or calculated.
3. Calculations performed only at the frequencies for which discrete data is available through either measurement or calculation.

There are some devices which are described accurately enough with analytical expressions over a limited frequency range. For example, smooth uniform transmission lines may be described with matrices whose entries contain hyperbolic functions over the

electrical length of the line. Lumped elements such as resistors, inductors, or capacitors may be characterized with the analytical expressions R, Ls, or $1/Cs$. Multiterminal lumped networks may be analyzed symbolically and characterized at given terminals with ratios of complex polynomials. Ideal elements such as bandpass filters with linear phase may be characterized by analytical phase and magnitude expressions. When such analytical expressions are available without too much additional effort and are accurate enough, they are preferable to other characterizations because of their convenience.

Sometimes analytical expressions are not available without considerable additional effort, but discrete data is. The discrete data may have been measured or calculated. If we know the general shape of the curve from which the discrete data points were sampled, the discrete data may be replaced by standard functions whose parameters are determined by a process of curve fitting [9]. The choice of standard functions is determined by the application. Some possibilities are

1. If the range of frequencies and the range of the values of the functions are small, low-order polynomials will generally be adequate for slowly varying functions.
2. If the range of frequencies is large and the range of values is small, low-order polynomials in the logarithm of the frequency are usually adequate.
3. If both the range of values and the range of frequencies are large, the logarithms of the values versus the logarithms of the frequencies may be fit with low-order polynomials.

If the data varies somewhat erratically, different curves may be fitted over each frequency range, for example, linear interpolation between strategically chosen data points.

The methods mentioned above are the simplest, and library routines are usually available to implement them. A user may, however, choose any functional form and adjust parameters with an optimization program if he so desires for some particular reason.

By having several subroutines available in a program, the user, depending on his problem, may through data cards control what type of standard functions are used for interpolation.

If the discrete data which is available is spaced sufficiently closely the analysis may be done only at the frequency values at which the data is available. This of course assumes that all the components described by discrete data are characterized at the same frequencies (or else that the analysis is done only at the frequencies for which all the elements are characterized). In this case curve fitting may be dispensed with altogether.

When approximating a given curve with a linear combination of a set of functions, there are certain problems that may arise. One of these problems is that calculations performed by computer carry a limited number of significant figures. Subtraction of large numbers to obtain small numbers may cause considerable error due to truncation. The situation will not arise if orthogonal functions are used in performing calculations by computer.

In function space the powers of x; $1, x, x^2, x^3, \ldots, x^k, \ldots$ are not orthonormal functions (orthogonal and normalized). The higher the order of the polynomials the less the projection of one upon another as exhibited by their inner product in the interval [0,1]

$$\langle x^p, x^q \rangle = \int_0^1 x^p \cdot x^q dx = \frac{x^{p+q+1}}{p+q+1}\Big]_0^1 = \frac{1}{p+q+1}$$

This means that when one fits a curve with a high-order polynomial, it will generally be necessary to carry a significant number of digits, especially for the higher powers. This is the reason for the following surprising fact: Although experimental data may be accurate to two significant figures, it may be necessary to provide polynomial coefficients which are accurate to seven figures. Unless one is aware of this one may be tempted to round off the polynomial coefficients to the same significant figures as the data. This would, of course, give very inaccurate results.

One way of alleviating this problem is to use functions other than powers of x. There exist several families of orthogonal polynomials which are orthonormal over different intervals [9]. Examples are: Legendre, orthogonal in $[-1,1]$; Chebyshev, First Kind, orthogonal in $[-1,1]$; Chebyshev, Second Kind, orthogonal in $[-1,1]$; Jacobi, orthogonal in $[-1,1]$; Generalized Laguerre, orthogonal in $[0,\infty]$; and Hermite, orthogonal in $[-\infty,\infty]$. These polynomials are solutions to certain differential or difference equations with particular boundary conditions. One may generate one's own orthogonal polynomials over any interval using the Gram-Schmidt process. More generally one may use linear combinations of any set of functions (preferably orthogonal but not necessarily so) to data-fit. The choice will be a compromise between availability of subroutines to do the fitting, size, efficiency, and

accuracy of the subroutines, and (quite important) the willingness of the electronic circuit designer to work with analytical tools which he is not accustomed to using.

One possibility which has been found quite effective and simple to use is piecewise fitting of curves over different intervals. Discrete points may be fed into the computer and different curves may be fit on different intervals. For instance, one approach is to fit a quadratic polynomial through the first three points and to subsequently fit other quadratics through each additional point, matching the previous curve both in value and derivative at the last common point. With the aid of IF statements the routine decides what quadratic to use depending on the interval to which the independent value belongs. In this manner a continuous curve with a continuous derivative passing through the data points is obtained. More sophistication is obviously possible. A simplified version of this approach is simple linear interpolation between data points. This yields a continuous curve between data points although the derivative will be discontinuous. Linear interpolation has been found so effective and simple to use that it is used as the standard option in BELNAP[1] with the other options available on request. It is particularly suitable for the characterization of curves which are used only once in the program and will not become part of the permanent library. When a subcircuit (such as certain transistors) is used very often and becomes part of the permanent library of the program, it is worth the effort to fit its curves with computationally efficient expressions [10]. Optimization programs such as SUPROX [11] have been found very useful in this respect. With an optimization program one assumes an expression with some variable parameters and the program automatically determines the best parameters for the fit, thus providing an analytical model for the device. A particular case of this is to fit lumped circuits to frequency curves.

4-3 CHARACTERIZATION OF CIRCUITS WITH THE INDEFINITE ADMITTANCE MATRIX

There are many matrices that can characterize a multiterminal network. Examples are impedance matrices, hybrid matrices, and scattering matrices. Because of its simplicity when handling interconnected black boxes, the indefinite admittance matrix [12] (IAM) will be one of the characterizing matrices in this chapter.

The IAM is the short-circuit admittance matrix of a multinode network in which ports are formed between each terminal and a datum node which is "floating," i.e., unconnected to the circuit. When connecting two-terminal loads to a multiterminal network it is sufficient to characterize it as a multiport with ports defined at each terminal pair to which a load is connected. However, when an n-terminal network is arbitrarily interconnected with other multiterminal networks it is necessary to characterize the network at a set of $n-1$ ports corresponding to a complete and independent set of terminal pairs. A sufficient condition for the voltages of the $n-1$ ports to be Kirchhoff-voltage-law-independent is that a graph made of edges representing the port voltages form a tree [13].

It is trivially simple to go back and forth between the indefinite admittance matrix and a (definite) short-circuit admittance matrix when all the ports have a common node which is connected to the circuit (usually referred to as ground) [14]. If all the ports do not have a common ground, however, the IAM is not obtained quite as simply. This makes it necessary to have a method of going from a given set of ports to a second set having a common node. This can be accomplished by using the following formula

$$Y = C^T \hat{Y} C \tag{4-1}$$

where Y is the admittance matrix with ports having a common node, $\hat{Y}$ is the original admittance matrix whose ports from a tree, C is the transpose of reduced incidence matrix of the graph whose edges represent the ports of $\hat{Y}$, and C^T is the transpose of C [13, 14].

The use of Eq. (4-1) can be illustrated with the following example: The circuit of Fig. 4-2(a) represents a pair of mutually coupled coils. Let port 1 be defined by making node 1 the positive terminal and node 2 the negative one, and port 2 with node 3 positive and node 4 negative. The open-circuit impedance matrix of this two-port is

$$\hat{Z}_1 = S \begin{bmatrix} L_{11} & L_{12} \\ L_{12} & L_{22} \end{bmatrix}$$

The inverse of $\hat{Z}_1$ is the admittance matrix with ports 1 and 2 defined as above

[1] A program to analyze black-box circuits which will be described below.

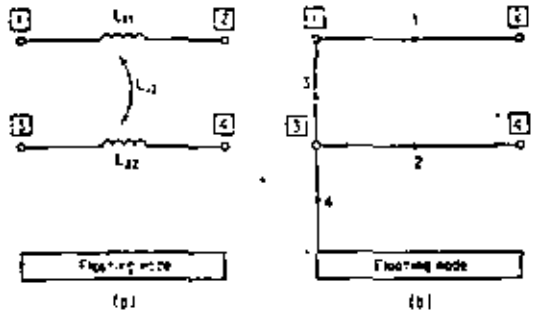


Fig. 4-2 (a) Two mutually-coupled inductors; (b) graph of the original four-port.

$$\hat{Y}_1 = \hat{Z}_1^{-1} = \frac{1}{s\left(L_{11}L_{22} - L_{12}^2\right)} \begin{bmatrix} L_{22} & -L_{12} \\ -L_{12} & L_{11} \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} \\ y_{12} & y_{22} \end{bmatrix}$$

To obtain the indefinite admittance matrix of the circuit of Fig. 4-2(a) it is necessary to obtain the short-circuit admittance matrix of the circuit having ports going from each terminal to a floating ground. To do this is is necessary to characterize the circuit at two additional ports. Let a third port be defined from nodes 3 to 1 in Fig. 4-2(a), and a fourth port from node 3 to the floating node. The admittance matrix of the circuit with the four ports as defined is

$$\hat{Y} = \begin{bmatrix} y_{11} & y_{12} & 0 & 0 \\ y_{12} & y_{12} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Now to apply Eq. (4-1), it is necessary to form the transpose of the reduced incidence matrix C of the graph defining the ports. This graph is shown in Fig. 4-2(b). The result is

$$C = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Thus Eq. (4-1) gives

$$Y = C^T \hat{Y} C = \begin{bmatrix} 1 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_{11} & y_{12} & 0 & 0 \\ y_{12} & y_{22} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$Y = \begin{bmatrix} y_{11} & -y_{11} & y_{12} & -y_{12} \\ -y_{11} & y_{11} & -y_{12} & y_{12} \\ y_{12} & -y_{12} & y_{22} & -y_{22} \\ -y_{12} & y_{12} & -y_{22} & y_{22} \end{bmatrix}$$

In similar fashion the indefinite admittance matrix of other typical components appearing in electronic circuits may be found. Some typical cases are presented compactly in Table 4-1.

Table 4-1 shows the indefinite admittance matrices of some typical devices when they are connected to the lowest-numbered nodes in a circuit (1, 2, 3, etc.). If a device is connected to nodes $i, j, k, \ldots$ instead of to 1, 2, 3, ..., then a simple replacement of indices i for 1, j for 2, k for 3, etc., gives the location of the entries to the corresponding indefinite admittance matrix. The total indefinite admittance matrix of a complicated circuit is simply obtained by subsequently adding in the proper positions the contributions of each of the devices. This avoids manipulating any topological matrices, since the indefinite admittance matrix is actually obtained by inspection. The entries of Table 4-1 should enable the reader to obtain by inspection the IAM of most linear transistor circuits appearing in practice. This topic is considered in more detail in the next section.

4-3.1 Analysis of Black-box Circuits Through the Indefinite Admittance Matrix

Since the indefinite admittance matrix (IAM) is the n-port short-circuit admittance matrix in which the ports are formed between each terminal and an unconnected node which is floating, the current going into each terminal may always be considered to

Table 4.1. Indefinite Admittance Matrices of Some Typical Subcircuits

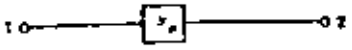
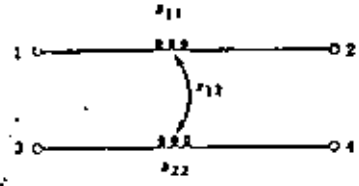
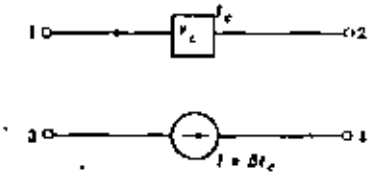
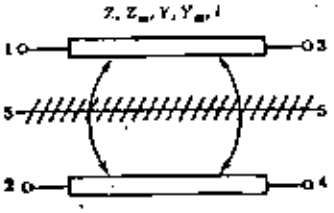
Circuit	Original $\bar{Y}$ matrix where ports form trees $\bar{Y}$	Final Y matrix when ports have common node Y
Element of admittance y_a connected between nodes 1 and 2  FLOATING NODE	PORT NODES 1 1, 2 2 1, F $\bar{Y} = \begin{bmatrix} y_a & 0 \\ 0 & 0 \end{bmatrix}$ $C = \begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix}$	PORT NODES 1 1, F 2 2, F $Y = \begin{bmatrix} y_a & -y_a \\ -y_a & y_a \end{bmatrix}$
Mutually coupled coils of self-impedances z_{11}, z_{22} and mutual impedance z_{12}  FLOATING NODE	PORT NODES 1 1, 2 2 3, 4 3 3, 1 4 3, F $\bar{Z}_1 = \begin{bmatrix} z_{11} & z_{12} \\ z_{12} & z_{22} \end{bmatrix}; \bar{Y}_1 = \bar{Z}_1^{-1}$ $\bar{Y} = \begin{bmatrix} y_{11} & y_{12} & 0 & 0 \\ y_{12} & y_{22} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	PORT NODES 1 1, F 2 2, F 3 3, F 4 4, F $Y = \begin{bmatrix} y_{11} & -y_{11} & y_{12} & -y_{12} \\ -y_{11} & y_{11} & -y_{12} & y_{12} \\ y_{12} & -y_{12} & y_{22} & -y_{22} \\ -y_{12} & y_{12} & -y_{22} & y_{22} \end{bmatrix}$
Current controlled current source Gain β controlled by current in y_c  FLOATING NODE	$C = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ PORT NODES 1 1, F 2 2, F 3 1, 2 4 3, 4 $\bar{Y} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & y_c & 0 \\ 0 & 0 & \beta y_c & 0 \end{bmatrix}$ $C = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$	PORT NODES 1 1, F 2 2, F 3 3, F 4 4, F $Y = \begin{bmatrix} y_c & -y_c & 0 & 0 \\ -y_c & y_c & 0 & 0 \\ \beta y_c & -\beta y_c & 0 & 0 \\ -\beta y_c & \beta y_c & 0 & 0 \end{bmatrix}$

Table 1.1. Indefinite Admittance Matrices of Some Typical Subcircuits (Continued)

Circuit	Original $\hat{Y}$ matrix where ports form tree $\hat{Y}$	Final Y matrix when ports have common node Y
Pair of identical, uniform coupled lines of longitudinal impedance Z, mutual impedance Z_m, admittance to ground Y, and admittance between lines Y_m, all per unit length. Length of lines l. [7]	PORT NODES 1 1,5 2 2,5 3 3,5 4 4,5 5 5,F	PORT NODES 1 1,F 2 2,F 3 3,F 4 4,F 5 5,F
Z, Z_m, Y, Y_m, l  FLOATING NODE	$\hat{Y} = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} & 0 \\ y_{12} & y_{11} & y_{14} & y_{13} & 0 \\ y_{13} & y_{14} & y_{11} & y_{12} & 0 \\ y_{14} & y_{13} & y_{12} & y_{11} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ $C = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$	$Y = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} & y_{15} \\ y_{12} & y_{11} & y_{14} & y_{13} & y_{25} \\ y_{13} & y_{14} & y_{11} & y_{12} & y_{35} \\ y_{14} & y_{13} & y_{12} & y_{11} & y_{45} \\ y_{51} & y_{52} & y_{53} & y_{54} & y_{55} \end{bmatrix}$

where

$$y_{11} = \frac{1}{2} M \coth(Pl) + \frac{1}{2} N \coth(Ql)$$

$$y_{12} = \frac{1}{2} M \coth(Pl) - \frac{1}{2} N \coth(Ql)$$

$$y_{13} = -\frac{1}{2} M \operatorname{csch}(Pl) - \frac{1}{2} N \operatorname{csch}(Ql)$$

$$y_{14} = -\frac{1}{2} M \operatorname{csch}(Pl) + \frac{1}{2} N \operatorname{csch}(Ql)$$

$$M = \sqrt{\frac{Y}{Z + Z_m}}, \quad N = \sqrt{\frac{Y + 2Y_m}{Z - Z_m}}, \quad P = \sqrt{(Z + Z_m)Y}, \quad Q = \sqrt{(Z - Z_m)(Y + 2Y_m)}$$

$$y_{15} = y_{25} = y_{35} = y_{45} = y_{51} = y_{52} = y_{53} = y_{54} = -(y_{13} + y_{12} + y_{13} + y_{14})$$

$$y_{55} = -(y_{15} + y_{25} + y_{35} + y_{45}) = 4(y_{11} + y_{12} + y_{13} + y_{14})$$

come out of the floating node under any type of interconnection and hence the condition for having a proper port is never lost. In the circuit of Fig. 4-3, the subnetworks N1, N2, and N3 can each be considered as five terminal networks, each one having one node dangling, and each having its own IAM. The IAM of the whole circuit is simply the sum of the indefinite admittance matrices of the subnetworks. This is very simple to program in a digital computer. All that is necessary is to know the contribution of each subnetwork separately. Each subnetwork may range from a simple resistor to a complicated integrated circuit or combinations of several of them.

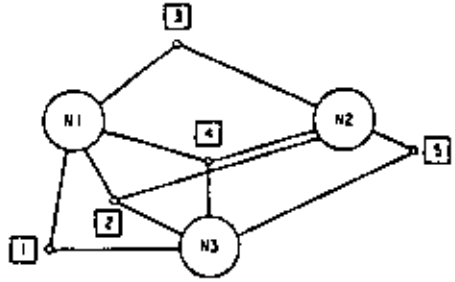


Fig. 4-3 Black-box network.

The principal properties of the indefinite admittance matrix are summarized here for convenience [12]

1. The sum of entries in each row is zero.
2. The sum of entries in each column is zero.
3. If the floating node is connected to the kth terminal of the circuit, the (definite) nodal matrix of the circuit with the kth terminal as datum is the indefinite admittance matrix with the kth row and the kth column deleted.
4. If the first p terminals of an n-terminal network are connected together, the new indefinite admittance matrix of the resultant $n-p+1$ terminal network is an $(n-p+1)\times(n-p+1)$ matrix obtained as follows: first form an intermediate matrix Q by substituting for the first p columns of the indefinite admittance matrix Y one column whose entries are the sums of the first p columns of Y. Next substitute for the first p rows of Q one row whose entries are the sums of the entries of the first p rows of Q. The resultant matrix is the new indefinite admittance matrix of the $n-p+1$ terminal network.

Once the indefinite admittance matrix of the circuit is established, it may happen that not all the nodes are of interest for forming ports. Some ports may be defined to simplify the modification of certain elements [15], or because the network will eventually be connected to other networks to form a still larger network. The nodes where ports will not be defined may be considered as internal nodes and suppressed.

The IAM is partitioned as in the following equation

$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \tag{4-2}$$

where I_1, V_1, are p-vectors, I_2, V_2 are $(n-p)$-vectors, Y_{11} is a $p\times p$ matrix, Y_{12} is a $p\times(n-p)$ matrix, Y_{21} is a $(n-p)\times p$ matrix, and Y_{22} is an $(n-p)\times(n-p)$ matrix. Assuming the first p terminals are the external terminals, $I_2 = 0$, since nothing will be connected to the internal terminals. Letting $I_2 = 0$ in Eq. (4-2) and solving the second line of Eq. (4-2) for V_2 in terms of V_1

$$V_2 = -Y_{22}^{-1} Y_{21} V_1 \tag{4-3}$$

which when substituted on the first line of Eq. (4-2) gives

$$I_1 = \left(Y_{11} - Y_{12}Y_{22}^{-1}Y_{21}\right) V_1 \tag{4-4}$$

Thus the suppressed admittance matrix Y_S is

$$Y_S = Y_{11} - Y_{12}Y_{22}^{-1}Y_{21} \tag{4-5}$$

The external node voltages may be solved for using Y_S, which is of smaller size than Y. Once the voltages V_1 of the external terminals are determined, the voltages of the internal terminals V_2 may be determined using Eq. (4-3).

It may happen that the internal nodes are connected to independent current sources. In this case I_2 is not zero when suppressing internal nodes in Eq. (4-2) but it has entries of values equal to the currents injected into each terminal by the independent current sources. In this case

$$V_2 = -Y_{22}^{-1}Y_{21}V_1 + Y_{22}^{-1}I_2 \tag{4-6}$$

which when substituted in the first line of Eq. (4-2) gives

$$I_1 = \left(Y_{11} - Y_{12}Y_{22}^{-1}Y_{21}\right)V_1 + Y_{12}Y_{22}^{-1}I_2 \tag{4-7}$$

which may be written

$$I_{eq} = \left(Y_{11} - Y_{12}Y_{22}^{-1}Y_{21}\right)V_1 \tag{4-8}$$

with

$$I_{eq} = I_1 - Y_{12}Y_{22}^{-1}I_2. \tag{4-9}$$

Equations (4-8) and (4-9) indicate that the internal current sources causing I_2 may be replaced by external currents $-Y_{12}Y_{22}^{-1}I_2$ and added to the original external currents I_1 to form an equivalent external current vector I_{eq}.

In order to analyze very large circuits, one may divide the circuit into pieces which may coincide with functional divisions. The IAM of each piece is obtained and the internal nodes are suppressed. All the previously suppressed black boxes may then be interconnected and a larger IAM including only external nodes is obtained and used to solve for all external terminal voltages. Finally in each subnetwork the internal voltages may be obtained. The saving comes from the fact that the internal nodes are not all in memory at once. This method is akin to Kron's method of tearing [16], though it does not invoke tensors. As presented here, it may include active devices and also distributed elements.

4-3.2 Calculation of Definite Matrices and Terminal Characteristics

Once the total suppressed IAM of a circuit is obtained, the matrix may be made definite by connecting one of the nodes (called ground) to the floating node and deleting the corresponding row and column to obtain the node-to-ground definite admittance matrix Y_{SG}. The node-to-ground matrix may be inverted to obtain the node-to-ground impedance matrix Z_{SG}. Since it may be desired to form ports other than node-to-ground, a transformation from node-to-ground ports to other ports is necessary. The formula to use is

$$Z = DZ_{SG}D^T \tag{4-10}$$

[a] In case independent voltage sources exist, Norton-equivalent independent current sources may replace them and the method applied to them.

where Z is the impedance matrix with the desired ports, and D is the transpose of the reduced incidence matrix of the graph corresponding to the desired ports, D^T is the transpose of D and Z_{SG} is the impedance matrix with ports defined from the nodes to ground [17].

Let the Z matrix of Eq. (4-10) have elements Z_{ij}

$$Z = (Z_{ij}) \tag{4-11}$$

Then the voltage ratios are given by

$$\frac{V_i}{V_j} = \frac{Z_{ii}}{Z_{ji}} \tag{4-12}$$

It is usually convenient to consider the loads as part of the network under analysis. Thus Z is the impedance matrix with the loads connected. Often it is of interest to find the driving-point impedance of a loaded circuit but excluding the generator impedance at the port looked into. To do this one may simply connect to the port in question an additional impedance equal to the negative of the generator impedance. Denoting with Z_{iii} the load impedance of the generator at port i, the driving point impedance z_i at port i is given by

$$z_i = \frac{1}{(1/Z_{ii}) - (1/Z_{iii})} \tag{4-13}$$

where Z_{ii} is the ith element on the main diagonal of the matrix Z of Eq. (4-10). The return loss RL_{ii} at port i is given in dB by

$$RL_{ii} = 20 \log \left| \frac{z_i + Z_{iii}}{z_i - Z_{iii}} \right| \tag{4-14}$$

The insertion voltage gain IG_{ij} between ports i and j is given in dB by

$$IG_{ij} = 20 \log \frac{(V_j/V_i)[z_i(z_i + Z_{iii})]}{[Z_{iii}/(Z_{ii} + Z_{iii})]} \tag{4-15}$$

4-3.3 Combining the Indefinite Admittance Matrix and Curve Modeling for Analysis of Integrated Circuits

A problem that commonly occurs in practice is the analysis of a circuit in which certain subcircuits recur. For example, a

circuit may contain several identical operational amplifiers. A convenient way of analyzing such circuits is the following: analyze each subcircuit separately and obtain its indefinite admittance matrix at several frequencies suppressing the internal terminals (terminals not connected to the rest of the network). Repeated subcircuits are analyzed only once. Model the subcircuits with polynomials fitting the frequency behavior of the real and imaginary parts of each entry in the admittance matrix. For example, if cubics are used for a circuit with four external terminals, 128 coefficients will model the device (4 (coefficients/cubic) times 4 × 4 × 2 (real and imaginary elements/matrix of 4 terminal network)). Interconnect the subcircuits to form a larger indefinite admittance matrix and again suppress the internal terminals. Repeat the process as required until the whole network is analyzed and the terminals of interest are the only external terminals.

Note that this method is essentially the same method one uses in designing large systems, where each subsystem is designed separately and eventually the different subsystems are connected. Several levels of subsystems may of course be used.

One of the important points to notice here is that a whole network which may have been analyzed in several levels of tearing may still be modeled with polynomials (or other standard interpolation methods) regardless of how complicated the internal structure of the network is. What we are trying to point out is that one can work advantageously with "universal" models which do not have to be the classical R, L, C, controlled-source, schematic models which network designers are accustomed to using.

In handling circuits with admittance matrices, certain degeneracies may occur for pieces of the network even though the whole circuit has a nondegenerate admittance matrix. In many cases these problems can be circumvented with special circuit techniques.

The indefinite admittance method is geared towards imbedded independent current sources rather than voltage sources. If the voltage source has an impedance in series, Norton-equivalent current sources may replace the voltage sources. If there is no impedance in series with the voltage source, one may add a positive and negative impedance of equal value in series and associate the voltage source with either to produce a Norton-equivalent current source. This method will create an additional node. Similar tricks may be employed for handling voltage-controlled voltage sources.

Certain multiterminal elements do not have an indefinite admittance matrix because infinite elements in the matrix are created due to what essentially amounts to short circuits between some terminals. One may avoid the degeneracy by using the trick of adding and subtracting elements in series. For example, perfectly coupled inductances and ideal transformers have no admittance matrix. However, by adding positive and negative resistances in series with each winding and considering the positive resistances as though they were associated with the windings and the negative resistances as elements separate from the device, the degeneracy disappears. This procedure may also be used for black boxes containing more terminals.

4-4 DESCRIPTION OF THE BELNAP PROGRAM

A computer program known as BELNAP (Bell Electronic Linear Network Analysis Program),[1] using the methods discussed above, has been written completely in FORTRAN IV and has been implemented on a GE-635 computer at Bell Telephone Laboratories [18, 19]. (A flowchart of the BELNAP appears in Table 4-2.)

The circuits which BELNAP can handle may include

1. ordinary positive and negative R, L, C, M elements
2. current-controlled current sources
3. distributed elements such as transmission lines, RC lines, coupled transmission lines
4. black boxes with n terminals which have been characterized at $n - 1$ independent ports. The characterization may be done with admittance matrices, impedance matrices, scattering matrices (for n-terminal networks) and A parameters or $ABCD$ parameters (for three-terminal networks)

The characterization of the black boxes may be done with tables of values at discrete frequencies, parameters of standard families of functions, or codes which call subroutines containing analytical models of the devices.

The program is limited to the following maxima: 40 nodes, 20 subnetworks of up to 8 ports each, 20 frequencies, 100 resistors, 100 inductors (25 mutuals), 100 capacitors, and 20 controlled current sources.

[1] BELNAP was written by L. A. Davison of Bell Telephone Laboratories.

Table 4-2. Flowchart of DELNAP

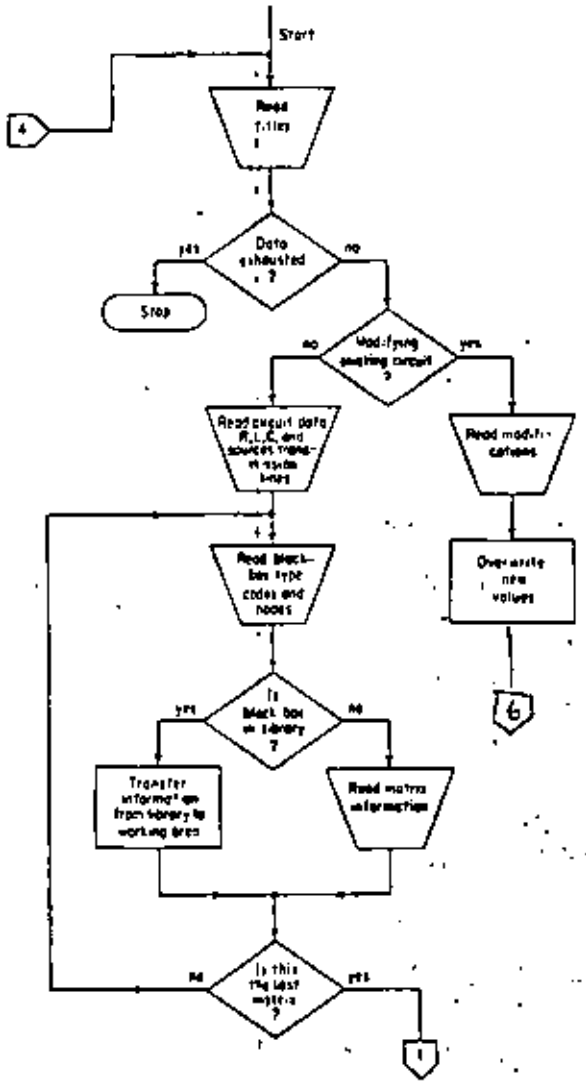


Table 4-2. Flowchart of DELNAP (Continued)

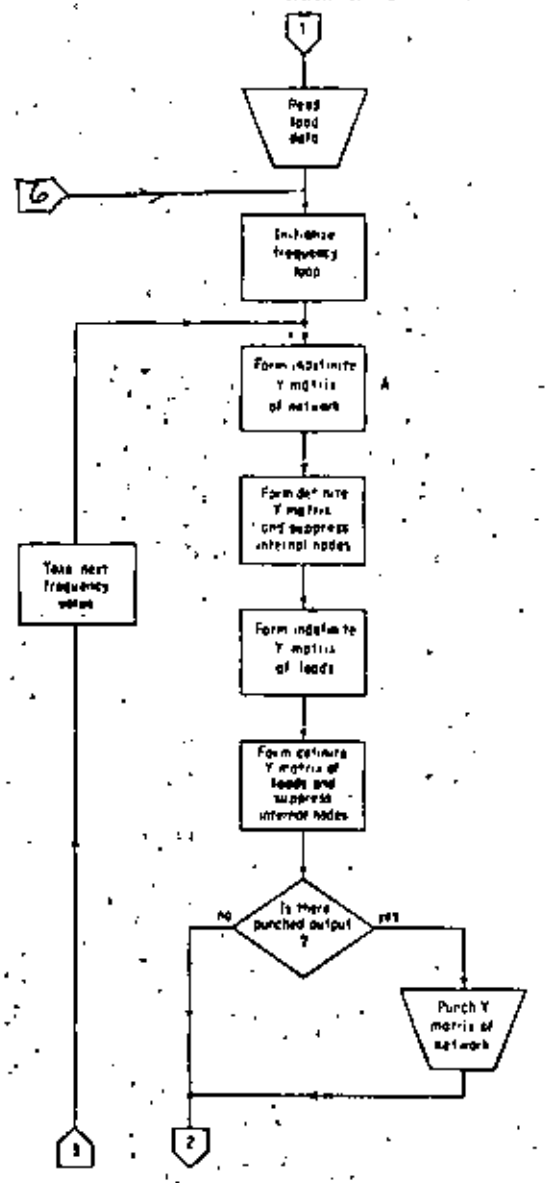


Table 4-2. Flowchart of BELNAP (Continued)

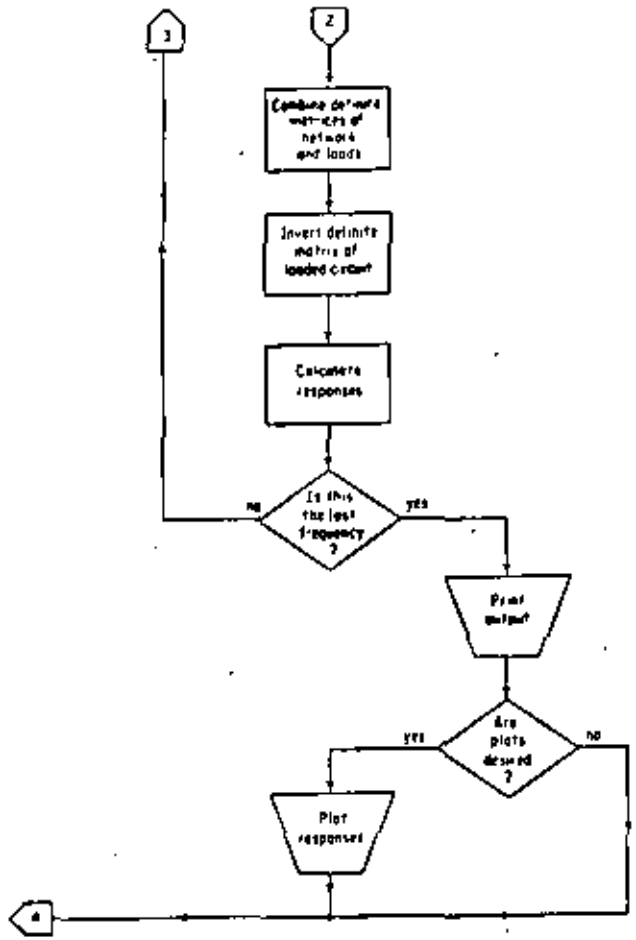
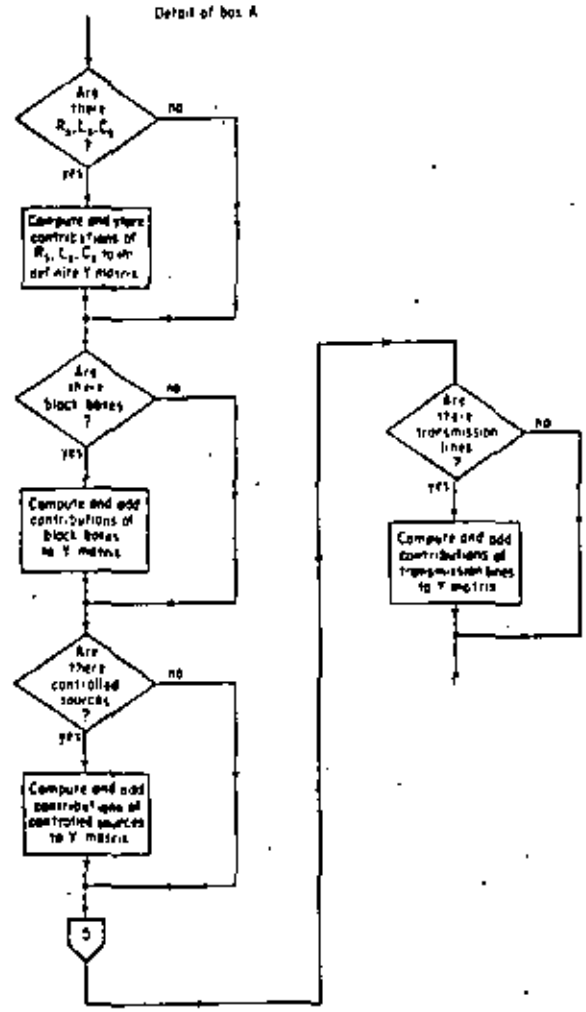


Table 4-2. Flowchart of BELNAP (Continued)



These limitations depend exclusively on memory requirements and can be changed by changing dimension statements. One of the versions used at Bell Laboratories has lower capabilities in order to be able to run in the express option for which it is required to use less than 40K of memory.

One of the features of the program is a library for commonly used black boxes. If a black box which is in the library appears in a circuit, all that is needed is to input it with a code name and to give the nodes to which it is connected in a given order.

A MODIFY option allows a user to change the values of R, L, C, or controlled-source betas and repeat the analysis without having to feed complete circuit descriptions. Several types of transmission lines can be specified by giving their characteristics. The program accepts discrete measured data in the form of admittance, impedance, or scattering $n \times n$ matrices and H parameters or $ABCD$ parameters for three-terminal networks. Values at intermediate frequencies are obtained automatically by linear interpolation.

The input to the program is user-oriented and in free format using the NAMELIST feature of FORTRAN IV. For the input of special elements, several choices exist, including, for example, coefficients for frequency power series or coefficients for the log of the function in terms of powers of the log of frequency.

The program computes the indefinite admittance matrix of the network at given discrete frequencies, suppresses the internal nodes indicated by the user, and punches on request the suppressed indefinite admittance matrix. It then inverts the reduced definite admittance matrix to obtain the nodal impedance matrix from which it obtains the desired driving point impedance at each port (excluding generator impedance), insertion voltage gains, voltage ratios, and return losses. The output may be obtained in either tabular form or in the form of frequency plots produced by a SC-4020 microfilm plotter.

4-4.1 Analysis of a Balanced Amplifier Using BELNAP

As an example of the kinds of problems that BELNAP may handle, the analysis of a balanced amplifier that contains distributed and lumped elements is given [20, 21]. The transistors have been characterized by actual measurements.

A general schematic of the amplifier is given in Fig. 4-4. The triangles represent identical transistor amplifiers and the square boxes represent stripline directional couplers. The schematic of a transistor amplifier appears in Fig. 4-5.

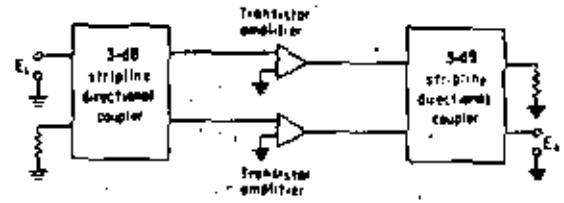


Fig. 4-4 Schematic of balanced amplifier containing lumped and distributed elements.

The analysis was done as follows: Certain portions of the network were preanalyzed or measured at a discrete set of frequencies and the resulting real parts and imaginary parts were fitted with standard functions.

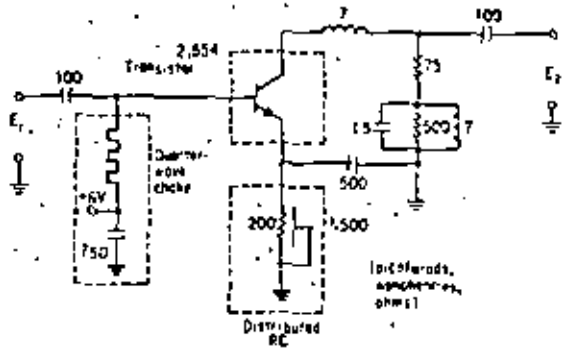


Fig. 4-5 Circuit schematic of a transistor amplifier.

In Fig. 4-5 the portions inside the dotted rectangles were precharacterized. The transistor was characterized by measuring its scattering matrix at discrete frequencies, converting to Y matrix, and fitting the discrete points with polynomials. The quarter-wave meander choke terminated in a capacitor was theoretically characterized at several discrete frequencies by calculations with an analytical expression that includes dissipation effects and the discrete values were fitted with standard functions.

The RC distributed circuit was modeled with 20 cascaded lumped RC sections which were analyzed and fitted with polynomials. (Although theoretical expressions could have been used, this method was chosen for illustrative purposes.) Once the elements in the dotted rectangles had been characterized as black boxes, the whole amplifier of Fig. 4-5 containing both lumped elements and black boxes was analyzed and characterized as a black box. Finally the whole amplifier of Fig. 4-4 was analyzed (the directional couplers having previously been analyzed with a special subroutine for coupled striplines) and the results printed and plotted [22].

A sample of the intermediate results are shown in Table 4-3 where a list of values of the real and imaginary part of y_{11} of the transistor of the amplifier for 10 discrete frequencies is shown and where the polynomials that were internally fitted to this data appear at the bottom of the table. This is also shown graphically in Figs. 4-10 and 4-11 where the rectangles give the measured data and the continuous curves give the points calculated with the

Table 4-3. Polynomial Fit of y_{11} of the GE-2554 Transistor

MHz	Experimental data		Points computed by polynomial	
	Real part	Imaginary part	Real part	Imaginary part
100	0.53×10^{-2}	0.253×10^{-2}	0.528×10^{-2}	0.254×10^{-2}
135	0.55×10^{-2}	0.34×10^{-2}	0.553×10^{-2}	0.344×10^{-2}
190	0.61×10^{-2}	0.46×10^{-2}	0.612×10^{-2}	0.480×10^{-2}
250	0.73×10^{-2}	0.60×10^{-2}	0.725×10^{-2}	0.581×10^{-2}
330	0.86×10^{-2}	0.71×10^{-2}	0.859×10^{-2}	0.707×10^{-2}
450	1.05×10^{-2}	0.825×10^{-2}	1.05×10^{-2}	0.836×10^{-2}
610	1.28×10^{-2}	0.93×10^{-2}	1.28×10^{-2}	0.941×10^{-2}
820	1.54×10^{-2}	1.00×10^{-2}	1.55×10^{-2}	0.978×10^{-2}
1100	1.85×10^{-2}	0.878×10^{-2}	1.84×10^{-2}	0.864×10^{-2}
1500	2.19×10^{-2}	0.54×10^{-2}	2.19×10^{-2}	0.539×10^{-2}

$$\mathrm{Re}\, y_{11} = 2.587270 \times 10^{-2} - 4.967302 \times 10^{-3}\phi - 4.316581 \times 10^{-4}\phi^2 - 2.752786 \times 10^{-5}\phi^3 + 4.545855 \times 10^{-5}\phi^4 - 3.048591 \times 10^{-5}\phi^5$$

$$\mathrm{Im}\, y_{11} = -1.670402 \times 10^{-1} + 8.671215 \times 10^{-2}\phi - 2.292190 \times 10^{-3}\phi^2 - 8.284316 \times 10^{-4}\phi^3 - 3.801567 \times 10^{-4}\phi^4 - 1.124385 \times 10^{-4}\phi^5 + 7.414405 \times 10^{-6}\phi^6$$

where $\phi = \log_{10} f$ and f is in MHz.

polynomials fitted. Figures 4-6, 4-7, and 4-8 show respectively the magnitude and phase of E_2/E_1 of the transistor amplifier of Fig. 4-5, the return loss at port 4 of the complete balanced amplifier of Fig. 4-4, and the magnitude and phase of the ratio of E_4/E_1 of the complete amplifier of Fig. 4-4 as produced by the microfilm plotter subroutine of BELNAP.

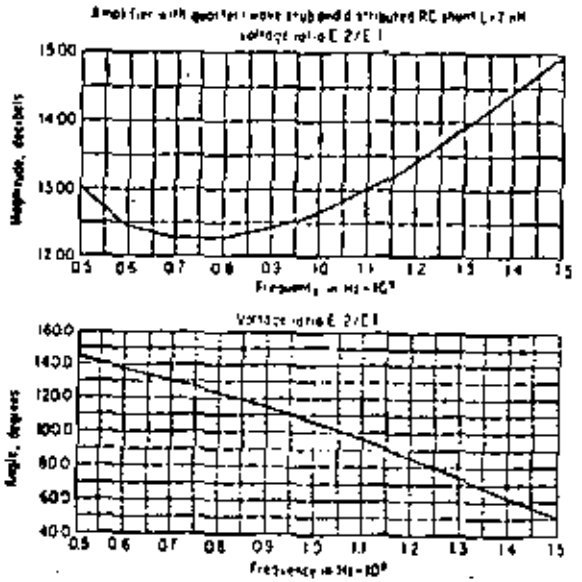


Fig. 4-6 Voltage ratio E_2/E_1 of a transistor amplifier.

4-4.2 BELTIP: Transient Version of BELNAP

Once the desired characteristics of a circuit are obtained in the complex frequency domain, it is possible to obtain them in the time domain by numerical Laplace transform inversion. Let $F(s)$ be the Laplace transform of a time function $f(t)$. The inverse transform formula is

$$f(t) = \frac{1}{2\pi j}\int_{s=\sigma-j\infty}^{\sigma+j\infty} F(s)e^{st}ds \qquad (4\text{–}16)$$

evaluated along a line to the right of the singularities of $F(s)$.

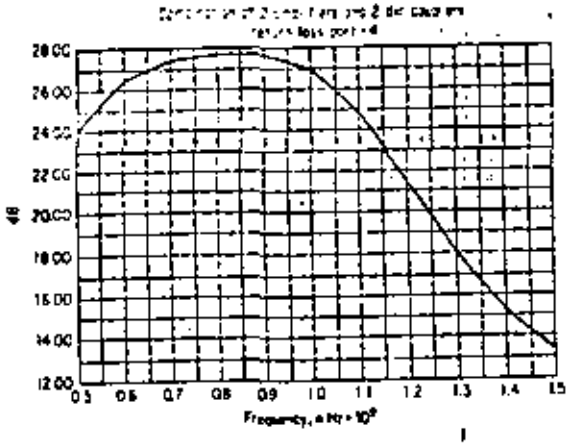


Fig. 4-7 Return loss at port 4 of the balanced amplifier.

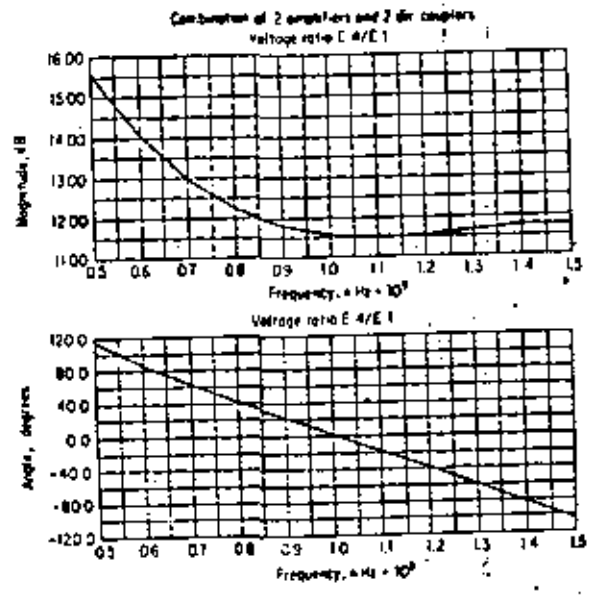


Fig. 4-8 Voltage ratio E_4/E_1 of the balanced amplifier.

The dummy integration variable s may be written $s = \sigma + j\omega$, with σ = constant. With the new integration variable ω, Eq. (4-16) reads

$$f(t) = \frac{1}{2\pi j}\int_{\omega=-\infty}^{\omega=\infty} F(\sigma + j\omega)e^{(\sigma+j\omega)t}\,j\,d\omega$$

from which

$$e^{-\sigma t}f(t) = \frac{1}{2\pi}\int_{\omega=-\infty}^{\omega=\infty} F(\sigma + j\omega)e^{j\omega t}\,d\omega \qquad (4\text{-}17)$$

which places in evidence the fact that $F(\sigma + j\omega)$ is the Fourier transform of the function $e^{-\sigma t}f(t)$.

Let us assume that $\int_{\omega=\omega_c}^{\infty}|F(\sigma + j\omega)|\,d\omega$ is negligible for some ω_c. For instance, this will be true if $|F(\sigma + j\omega)|$ goes down as $1/\omega^2$ nears infinity. Let us also assume that $F(\sigma + j\omega)e^{j\omega t}$ varies slowly enough with ω so that it may be approximated by a series of piecewise complex constants of values $F(\sigma + jk\Delta\omega)e^{jk\Delta\omega t}$, $k = 0, \pm 1, \pm 2, \ldots, \pm(N-1)/2$; $\Delta\omega = 2\omega_c/N$; N odd; for equal intervals of length $\Delta\omega$.

The integral of Eq. (4-17) may be approximated with a sum as follows

$$\frac{1}{2\pi}\int_{\omega=-\omega_c}^{\omega=\omega_c} F(\sigma + j\omega)e^{j\omega t}d\omega = \frac{\Delta\omega}{2\pi}\sum_{k=-(N-1)/2}^{(N-1)/2} F(\sigma + jk\Delta\omega)e^{jk\Delta\omega t} \qquad (4\text{-}18)$$

Now note that the last sum may be interpreted as a finite exponential Fourier series whose coefficients are $(\Delta\omega/2\pi)F(\sigma + jk\Delta\omega)$. The time function which the sum of Eq. (4-18) adds up to is a periodic function of period $T = 2\pi/\Delta\omega$. If the quantity $\Delta\omega$ is small, the function will repeat itself after a long period of time T. In the limit as $\Delta\omega \to 0$, $N \to \infty$, $T \to \infty$, $\omega_c \to \infty$, the function ceases to repeat itself (since the period is infinite) and the approximate sum of Eq. (4-18) gives the exact value of the integral on the left.

There are three parameters to be chosen, σ, ω_c, and N. Their choice is made according to the following considerations. The larger the σ the better the function $F(\sigma + j\omega)e^{j\omega t}$ can be approximated piecewise by constants since the function will be evaluated far from the singularities. On the other hand, the time function obtained has to be multiplied at the end by $e^{\sigma t}$; any errors at large values of time are thus exaggerated. The cutoff frequency ω_c

should be such that

$$\epsilon = \frac{1}{\pi} \int_{-\infty}^{-\sigma_a} |F(\sigma + j\omega)| \, d\sigma \tag{4-19}$$

is small. The quantity ϵ gives a bound on the maximum error at any time due to truncation of the spectrum in calculating $e^{-\sigma t} f(t)$. The number N is chosen so that the function $F(\sigma + j\omega)e^{j\omega t}$ remains reasonably constant during the intervals of length $\Delta\omega = 2\pi_c/N$. If the function varies rapidly, N will have to be large so that the intervals of length $\Delta\omega$ be small.

A sum of the form of Eq. (4-18) is evaluated using a modification of the Cooley-Tukey algorithm [23, 24] for fast complex Fourier series.[1] The result is a function which approximates $e^{-\sigma t} f(t)$ which when multiplied by $e^{\sigma t}$ gives the desired time function $f(t)$.

Figure 4-9 shows a time plot produced by BELTIP (Bell Electronic Laplace Transform Inversion Program).[2]

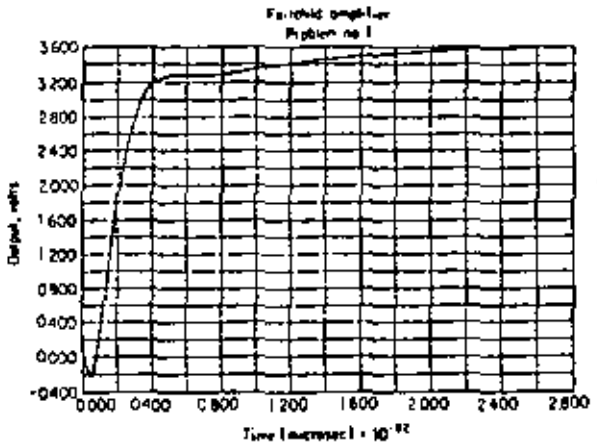


Fig. 4-9 Time response of an amplifier to a step.

When a circuit is being analyzed for time domain response, it is necessary to be able to obtain the frequency response for all the

[1] The Cooley-Tukey algorithm requires N to be a power of 2. Thus the discussion is intended as a guide for understanding the source of errors of the approximation.

[2] BELTIP was developed by M. Silverberg of Bell Telephone Laboratories.

frequencies for which its magnitude is not negligible and it may in some cases be necessary (depending on the location of the singularities) to be able to obtain such frequency information not at the $j\omega$ axis but on the complex plane. Thus, during fitting of standard functions these facts should be borne in mind. It has been noted, for example, that unless the real and imaginary part satisfies the Hilbert transform condition [14], the resultant computations yield highly anticipatory circuits which respond before being excited.

4-5 ANALYSIS OF CIRCUITS USING THE INDEFINITE TRANSFER MATRIX

Given a $2n$-terminal network, the indefinite transfer matrix E is defined by the following equations

$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \begin{bmatrix} V_2 \\ -I_2 \end{bmatrix}, \quad E = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \tag{4-20}$$

where V_1, I_1, V_2, I_2 are n entry columns corresponding to the currents and voltages shown in Fig. 4-11 in which the ports $1, 2, \ldots, n$ are considered input ports and the ports $n+1, n+2, \ldots, 2n$ are considered output ports. The matrices A, B, C, D are $n \times n$ matrices. (Their names result from the fact that they are extensions to $2n$-ports of the familiar A, B, C, D parameters of two-ports.) Like the indefinite admittance matrix, all ports are defined with a common floating terminal. For this reason the port condition is never lost with arbitrary interconnections. It is not difficult to go from the indefinite admittance matrix to the indefinite transfer matrix and vice versa. Let us partition the indefinite admittance matrix of a $2n$-terminal network as follows

$$Y = \left[\begin{array}{c|c} Y_{11} & Y_{12} \\ \hline Y_{21} & Y_{22} \end{array}\right] \tag{4-21}$$

The matrices $Y_{11}, Y_{12}, Y_{21}, Y_{22}$ are $n \times n$ matrices. The variables are ordered so that Y_{11} correspond to V_1 and I_1 of Eq. (4-20). The equations relating the Y of Eq. (4-21) with A, B, C, D of Eq. (4-20) are

$$Y_{11} = DB^{-1}, \; Y_{21} = C - DB^{-1}A, \; Y_{21} = -B^{-1}, \; Y_{22} = B^{-1}A \tag{4-22}$$

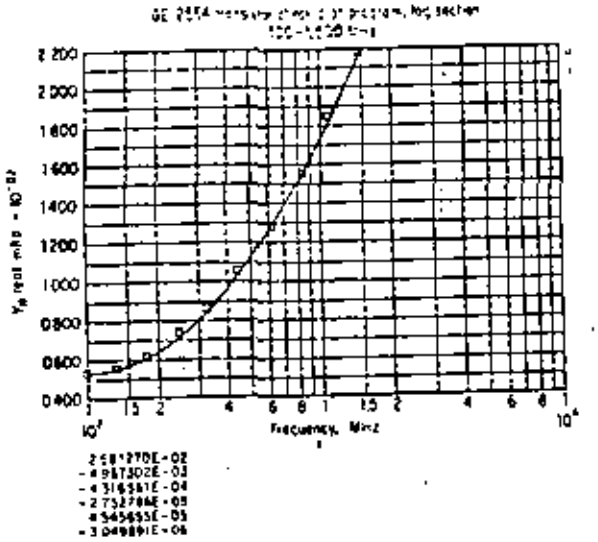


Fig. 4-10 Polynomial fit to $R_e\, y_{11}$ for GE-2554 transistor.

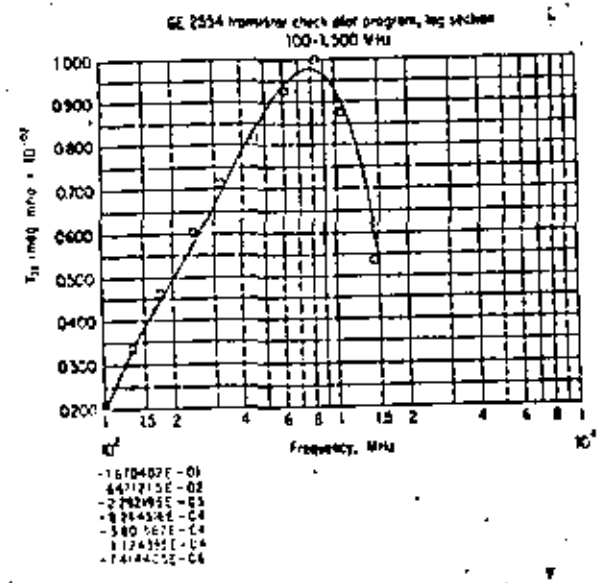


Fig. 4-11 Polynomial fit to $\mathrm{Im}\, y_{11}$ for GE-2554 transistor.

$$A = -Y_{21}^{-1}Y_{22},\quad B = -Y_{21}^{-1},\quad C = Y_{12} - Y_{11}Y_{21}^{-1}Y_{22},\quad D = -Y_{11}Y_{21}^{-1} \tag{4-23}$$

The most useful property of the E matrix is that if several two-terminal networks are connected in cascade the E matrix of the network is equal to the product of the individual E matrices of the cascaded sections. Because all the ports have a common terminal the port condition is never lost and no ideal transformers or tests for circulating currents are necessary [12].

In the analysis of electric circuits with a digital computer one often sacrifices computational efficiency to gain generality when using a "general purpose analysis program." When analysis and optimization programs are coupled, or in the making of statistical variability studies, it may be necessary to use the analysis program hundreds of times before the design process is over; thus the computational efficiency becomes increasingly important for such cases.

To obtain absolute maximum computational efficiency it would be necessary to write specific programs (perhaps in basic languages) for each circuit to be analyzed. Each program would be tailored to the peculiarities of the circuit in order to avoid the performance of any unnecessary operations. This is quite laborious, especially for large circuits. If many units of a particular circuit are going to be manufactured and hence exhaustive computer studies are going to be made, the writing of particular programs for circuits may be economically justified. If, on the other hand, many circuits which are members of a family are to be manufactured, although no particular member has a large volume of production, then it is better to write a slightly more general program that is capable of analyzing the whole family and would still be more efficient than a general purpose analysis program.

We will exhibit a family of transmission circuits for which extensive Monte Carlo statistical variability studies were required. It is possible to analyze the family of circuits with the indefinite transfer matrix and thereby gain enormous computational advantage over methods such as the indefinite admittance matrix which require matrix inversion.

Many four-ports which are of interest in radar and other communication systems fall into the class shown schematically in Fig. 4-12. The boxes labeled $N_1, N_2, \ldots, N_n$ represent four-terminal circuits plus a "ground." A port is made from each terminal to ground. Each four-port may be characterized by a 4×4 transmission matrix E each of which is partitioned into A, B, C, D matrices which are 2×2. All matrices have entries which are complex numbers and depend on frequency.

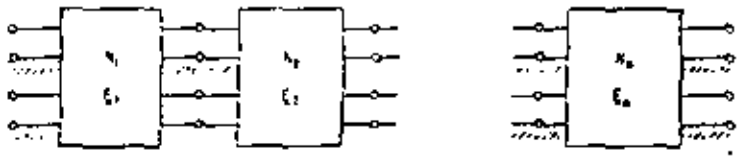


Fig. 4-12 Cascaded four-ports which are of interest in some communication systems.

To obtain the total E matrix of the n cascaded four-ports, the individual E matrices of the sections are multiplied in the order in which they appear. E_T denotes the E matrix of the complete network and E_k the E matrix of the kth section

$$E_T = E_1 \cdot E_2 \cdots E_k \cdots E_n \tag{4-24}$$

The first and last sections may correspond to the driving and load networks.

Each of the subnetworks of Fig. 4-12 may have an arbitrary internal structure. The characterization of each subnetwork may be done theoretically or experimentally. Curve fitting methods and preanalysis of subcircuits are also applicable here. Certain configurations which have application in the realization of directional couplers will be considered in detail. Figure 4-13 shows two identical lossless coupled transmission lines each with a capacitance to ground per unit length C, capacitance between lines per unit length C_M, self-inductance per unit length L_{11}, and mutual inductance per unit length L_{12}. The length of the lines is l. The E matrix of the four-port of Fig. 4-13 is

$$E = \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \left[\begin{array}{c|c} \cosh \Gamma \cdot l & (\sinh \Gamma \cdot l) Z_0 \\ \hline Y_0 \sinh \Gamma \cdot l & \cosh \Gamma \cdot l \end{array}\right] \tag{4-25}$$

where Γ, Z_0, and Y_0 are 2×2 matrices given by

$$\Gamma = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} j\omega \sqrt{(L_{11} + L_{12})C} + \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} j\omega \sqrt{(L_{11} - L_{12})(C + 2C_M)} \tag{4-26}$$

Fig. 4-13 Pair of lossless coupled lines.

$$Z_0 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \sqrt{\frac{L_{11} + L_{12}}{C}} + \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \sqrt{\frac{L_{11} - L_{12}}{C + 2C_M}} \tag{4-27}$$

$$Y_0 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \sqrt{\frac{C}{L_{11} + L_{12}}} + \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \sqrt{\frac{C + 2C_M}{L_{11} - L_{12}}} \tag{4-28}$$

The functions $\Gamma \cdot l$ and $\sinh \Gamma \cdot l$ of the matrix Γ are the following matrices

$$\cosh \Gamma \cdot l = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \cosh \left\{ \sqrt{(L_{11} + L_{12})C}\ j\omega l \right\} + \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \cosh \left\{ \sqrt{(L_{11} - L_{12})(C + 2C_M)}\ j\omega l \right\} \tag{4-29}$$

$$\sinh \Gamma \cdot l = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \sinh \left\{ \sqrt{(L_{11} + L_{12})C}\ j\omega l \right\}$$

$$+ \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \sinh \left\{ \sqrt{(L_{11} - L_{12})(C + 2C_M)}\ j\omega l \right\} \qquad (4\text{–}30)$$

Besides the distributed section considered above, two lumped subnetworks will be considered.

The first lumped subnetwork to be considered is the one shown in Fig. 4-14. The E matrix of the circuit of Fig. 4-14 is

$$E = \left[\begin{array}{c|c} I & Z \\ \hline 0 & I \end{array}\right] \qquad (4\text{–}31)$$

where 0 is the 2 × 2 zero matrix, I is the 2 × 2 unit matrix, and Z is

$$Z = \begin{bmatrix} j\omega L_{11} + R_{11} & j\omega L_{12} \\ j\omega L_{12} & j\omega L_{11} + R_{22} \end{bmatrix} \qquad (4\text{–}32)$$

The second type of lumped subnetwork to be considered is shown in Fig. 4-15 where Y_a, Y_b, and Y_c are the admittances of the boxes. The E matrix of the subnetwork of Fig. 4-15 is

$$E = \left[\begin{array}{c|c} I & 0 \\ \hline Y & I \end{array}\right] \qquad (4\text{–}33)$$

where Y is

$$Y = \begin{bmatrix} Y_a + Y_c & -Y_c \\ -Y_c & Y_b + Y_c \end{bmatrix} \qquad (4\text{–}34)$$

For the particular cases of Fig. 4-16 and 4-17, the corresponding Y matrices are

$$Y = \begin{bmatrix} (C_1 + C_3)j\omega + G_1 + G_3 & -C_3 j\omega - G_3 \\ -C_3 j\omega - G_3 & (C_2 + C_3)j\omega + G_2 + G_3 \end{bmatrix} \qquad (4\text{–}35)$$

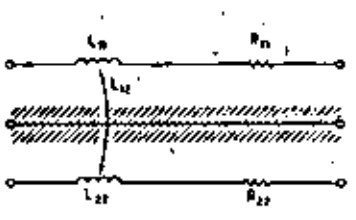


Fig. 4-14 Pair of lossy coupled coils forming a section of the cascade.

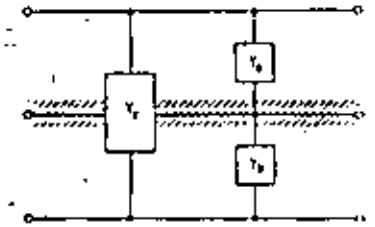


Fig. 4-15 Transversal admittances forming a section of the cascade.

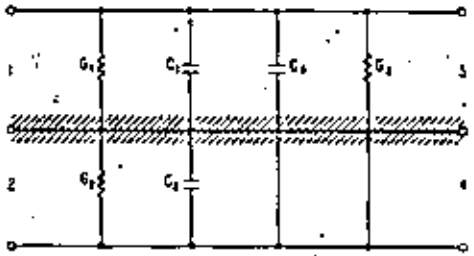


Fig. 4-16 Particular case of the section of Fig. 4-15.

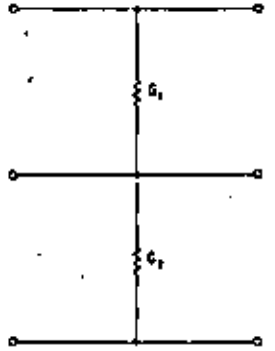


Fig. 4-17 Resistive loads considered as a particular case of Fig. 4-15.

$$Y = \begin{bmatrix} G_1 & 0 \\ 0 & G_2 \end{bmatrix} \tag{4-36}$$

where G_i are conductances in mhos and C_i are capacitances in farads.

Once the E matrix of the complete circuit or of a portion of it is known the voltage ratios are calculated very simply in terms of the entries of E. Partitioning the E matrix into 2×2 matrices A, B, C, D and assuming that a four-port is driven at port 1, we have

$$\frac{V_3}{V_1} = \frac{C_{22}}{C_{22}A_{11} - C_{21}A_{12}} \tag{4-37}$$

$$\frac{V_4}{V_1} = \frac{-C_{21}}{C_{22}A_{11} - C_{21}A_{12}} \tag{4-38}$$

$$\frac{V_2}{V_1} = \frac{C_{22}A_{21} - C_{21}A_{22}}{C_{22}A_{11} - C_{21}A_{12}} \tag{4-39}$$

The driving-point impedance at port 1 is

$$z = \frac{A_{11}C_{22} - A_{12}C_{21}}{C_{11}C_{22} - C_{12}C_{21}} \tag{4-40}$$

The reflection coefficient Γ_R when port 1 is connected as a load to a lossless transmission line of characteristic impedance z_0 is

$$\Gamma_R = \frac{z - z_0}{z + z_0} \tag{4-41}$$

Γ_R will in general be complex. Denoting its magnitude with $|\Gamma_R|$, the voltage standing wave ratio is given by

$$\text{VSWR} = \frac{1 + |\Gamma_R|}{1 - |\Gamma_R|} \tag{4-42}$$

With these tools, the analysis of a circuit containing lumped and distributed parameters, such as the one shown in Fig. 4-18, is easily done by matrix multiplication and application of Eqs. (4-37) to (4-42).

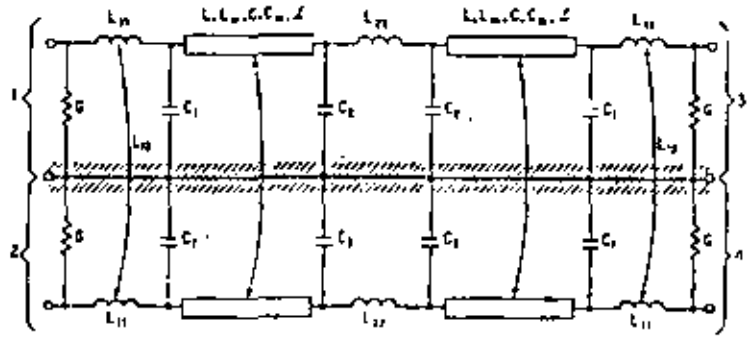


Fig. 4-18 Typical circuit that can be analyzed by COPLER.

Suppose that a circuit of the form of Fig. 4-12 is made up of sections each of which has the internal structure of the circuit shown in Fig. 4-19. Suppose the complete circuit has 10 sections. Each section has 15 nodes. The complete circuit will have 134 nodes (some of nodes are shared by two sections). A program such as ECAP would not be able to accommodate such a circuit and even

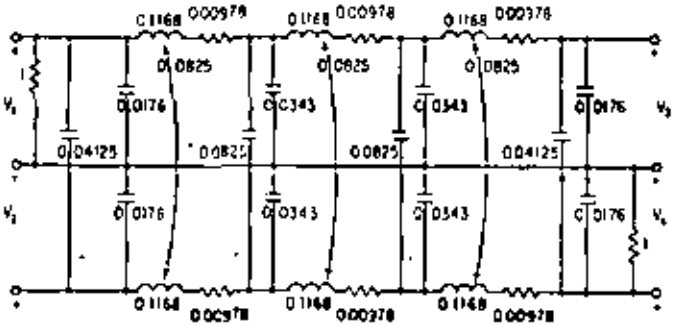


Fig. 4-19 Lumped circuit forming a section of the circuit of Fig. 4-12.

if it were, the amount of computation to solve the circuit would be considerable. Using an algorithm such as Gauss' elimination procedure, it takes roughly $(1/3)n^3$ complex multiplication-additions to solve the system of equations at each frequency. Here n stands for the order of the matrix which corresponds to the number of nodes minus one. Thus the circuit we are considering would require roughly $1/3\,(133)^3 = 0.8$ million multiplication-additions to solve for one frequency.

The nodal admittance matrix of the network we are considering is quite sparce (has many zeros); there are more efficient methods of solving it if its sparsity is exploited.

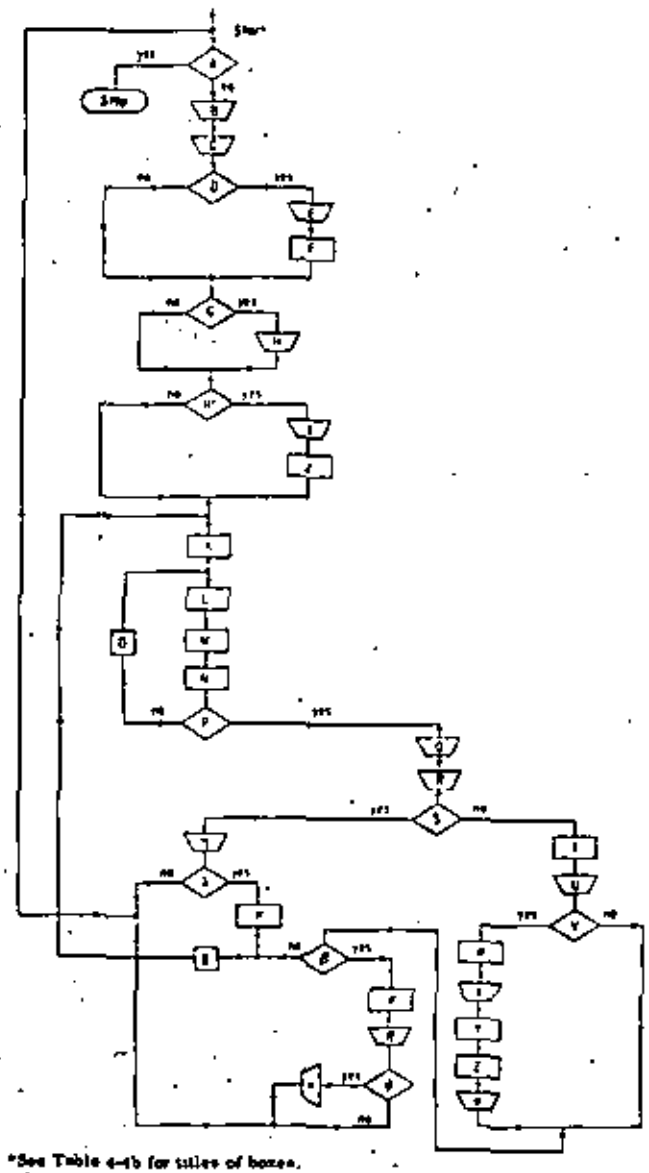
If the transfer matrix method is used and the network is considered as a cascade connection of four-ports of the types discussed above, the number of complex multiplication-additions per frequency is 5760, that is, only 1/130 as many.[1] Some Monte Carlo runs in which each run involved 500 circuits calculated at 10 frequencies were performed. The reader can estimate the enormous savings that were realized by using this method instead of the nodal matrix method. The principal reason for the savings is that with cascade configurations the number of operations to solve the circuit is a linear function of the number of nodes as opposed to a cubic function for general configurations and using the nodal matrix. Even if the transfer matrix were not used, the problem could have been solved much more efficiently by considering each section as a subnetwork, analyzing each subnetwork, and finally connecting all the sections obtaining an admittance matrix of order 20 which is considerably smaller than 133 [25].

4-6 BRIEF DESCRIPTION OF COPLER: A FOUR-PORT SIMULATOR

COPLER is a computer program written at Bell Telephone Laboratories [26] for simulating a class of four-ports. (A flowchart of COPLER is given in Table 4-4(a), and a description of this block appears in Table 4-4(b).) The program has been used mainly to predict the behavior of lumped and/or distributed directional couplers [27]. The program calculates the total E matrix of a circuit of the form of Fig. 4-12 given the values of the lumped R, L, C, M elements, values per unit length of coupled lines, or special

[1] Assuming that all the elements have different values. If the cascaded sections are equal, the savings are even larger.

Table 4-4a. Flowchart of COPLER Program*



*See Table 4-4b for tables of boxes.

Table 4-4b. Titles of Boxes for COPLER Program

A	Is data submitted?
B	Read titles, configuration code, and option flags.
C	Read element values for standard subcircuits.
D	Is statistical study desired?
E	Read statistical parameters and number of trials.
F	Store nominal parameters.
G	Is optimization desired?
H	Read element values and type of criterion of performance.
H′	Are there special black boxes?
I	Read portion of special black boxes.
J	Call subroutines for computing $\mathcal{E}$ matrices of special black boxes.
K	Initialize frequency loop.
L	Compute $\mathcal{E}$ matrices of standard subcircuits.
M	Multiply $\mathcal{E}$ matrices of complete cascade.
N	Calculate voltage ratios and VSWR.
O	Take next frequency.
P	Is this the last frequency?
Q	Print table of voltage ratios and VSWR versus frequency.
R	Print present values of parameters and trial.
S	Is this the nominal trial?
T	Calculate and store performance index.
U	Print performance index and trial number.
V	Is optimization desired?
W	Calculate criterion of performance.
X	Print criterion of performance and trial number.
Y	Store best criterion and trial number so far.
Z	Store table of voltage ratios and VSWR of best trial so far.
α	Print best criterion of performance and trial number so far.
β	Have enough statistical trials been calculated?
σ	Calculate moments of performance index.
θ	Print moments of performance index.
ξ	Make random variation of parameters around nominal values.
η	Plot the voltage ratios and VSWR.
λ	Is statistical study or optimization desired?
κ	Store nominal voltage ratios and VSWR.
φ	Is optimization desired?
ψ	Plot voltage ratios and VSWR of nominal and best trials.

subroutines defining arbitrary analytical models of four-ports in the frequency domain. A code vector indicates to the program how the elements are connected. The program also performs a statistical tolerance analysis using a Monte Carlo method and taking advantage of the tolerance analysis optimizes the network through a random walk. In order to do the statistical variability study COPLER has several pseudo random number generating subroutines. Taking advantage of the fact that the circuit is analyzed for the statistical variability study, optimum parameters for the system may be found by calculating a "criterion of performance" for each variation of the circuit and using the values of the parameters that give the best criterion of performance.

The output of COPLER under various options includes: printing or printing and plotting the phase and magnitude of the voltage ratios given by Eqs. (4-37)–(4-39) and the VSWR given by Eq. (4-42); printing of a "performance index" for each trial, and the first and second moments of the "performance index" of a set of random trials around a nominal network (the random parameters have specified statistical distributions). For each trial the program also prints the complete set of parameters of the circuit analyzed. Given a desired frequency behavior the program calculates a "criterion of performance" and prints the set of parameters with the best criterion of performance. There are several criteria of performance available and the user may define his own by writing his own subroutine.

The input to the program is user-oriented. The data is fed in formatless form using the NAMELIST feature of FORTRAN IV. If many elements are repeated there are simple ways of feeding them in. The circuit of Fig. 4-19 with the initial normalized values shown was analyzed with the aid of COPLER. Figure 4-20 shows the magnitude of V_3/V and Fig. 4-21 the VSWR versus normalized frequency of three random variations of the circuit of Fig. 4-19. Figure 4-22 shows a plot produced by the SC-4020. The plot corresponds to the circuit of Fig. 4-18 with the following normalized values: $L_{11} = L_{22} = 0.03533$ henries, $L_{12} = 0.025$ henries, $C_1 = C_2 = 0.01033$ farads, $G = 1$ mho, $L = 0.318$ henries/meter, $L_m = 0.225$ henries/meter, $C = 0.093$ farads/meter, $C_m = 0.225$ farads/meter.

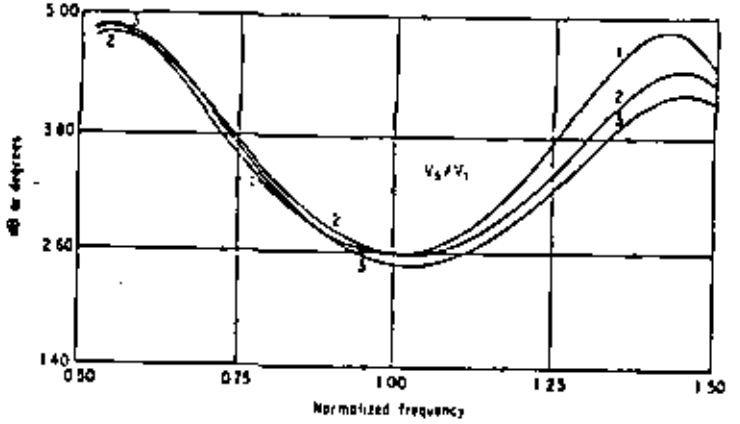


Fig. 4-20 Graph of $|V_3/V_1|$ of three random variations of the circuit of Fig. 4-18.

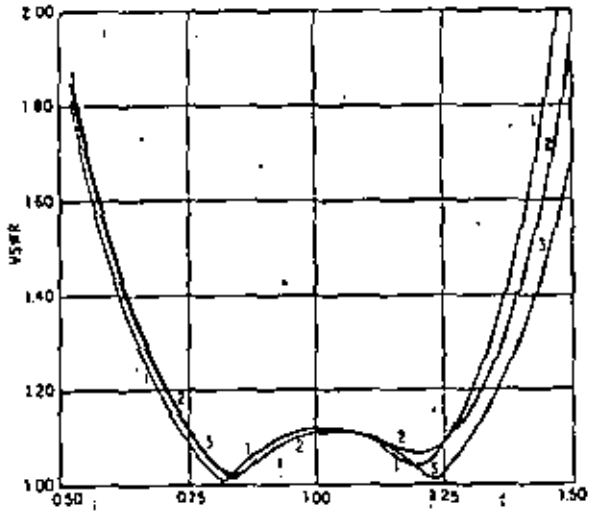


Fig. 4-21 Graph of the VSWR of three random variations of the circuit of Fig. 4-18.

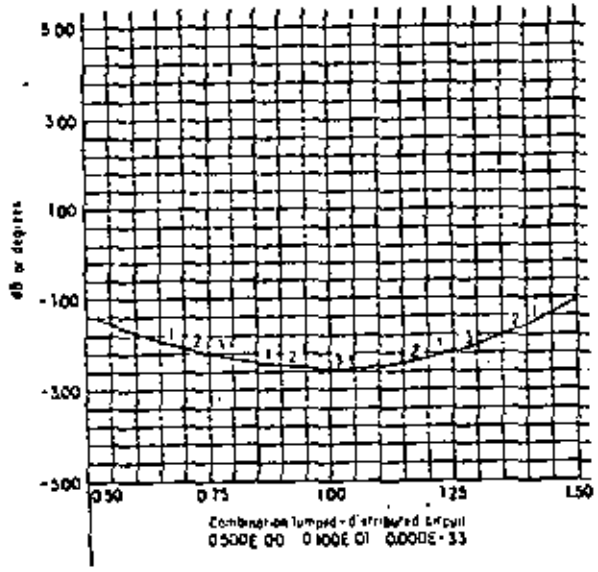


Fig. 4-22 Graph of $|V_3/V_1|$ of a lumped-distributed circuit of the form of the one in Fig. 4-19.

and $l = 0.33$ meter. The plot shows the magnitude in dB of the voltage ratio V_3/V_1 versus normalized frequency.

4-7 OUTLOOK

In this chapter the analysis of linear circuits consisting of interconnected black boxes was treated. (We consider conventional lumped elements as particular cases of black boxes.) Three programs, BELNAP, BELTIP, and COPLER, were briefly described and examples of their output were given. The black-box approach to the analysis of integrated circuits offers several advantages not available with the existing general-purpose computer programs. Among these advantages are: analysis of large circuits by pieces, modeling of transistors and integrated circuits by experimental measurements, and inclusion of distributed elements. The analysis of circuits by pieces not only alleviates the problem of limited memory, but also gains considerable computational efficiency since it takes advantage of the sparsity of the matrices of the networks.

REFERENCES

1. 1620 Electronic Circuit Analysis Program (ECAP), "Application Program 1620-EE-02X," Data Processing Division, IBM Corporation, White Plains, N. Y.
2. Malmberg, A. F., F. L. Cornwell, and F. N. Hofer: NET-1 Network Analysis Program, *Report LA-3119*, Los Alamos Scientific Laboratory, Los Alamos, N. M., 1964.
3. Happ, W. W.: NASAP: Present Capabilities of a Maintained Program, *Proc. Conf. Analysis of Circuits with Digital Computer*, Nat. Univ. Mexico, Mexico City, June, 1967.
4. Dickhaut, R. D.: CIRCUS, a Digital Computer Program for Transient Analysis of Electronic Circuits, Computer-Aided Circuit Design Seminar, M.I.T., Cambridge, Mass., seminar proc. published by NASA/ERC, pp. 4-143, April, 1967.
5. Sedore, S. R.: SCEPTRE: A Second Generation Transient Analysis Program, Computer-aided Circuit Design Seminar, M.I.T., Cambridge, Mass., seminar proc. published by NASA/ERC, pp. 57-61, April, 1967.
6. Automated Digital Computer Program for Determining Responses of Electronic Systems to Transient Nuclear Radiation, vol. II, File No. 64-521-5, IBM Corporation, Owego, N. Y., July, 1964.

7. Searle, C. L., et al.: "Elementary Circuit Properties of Transistors," pp. 102-120. John Wiley & Sons, Inc., New York, 1964.
8. Carlin, H. J.: Network Theory without Circuit Elements, *Proc. IEEE*, vol. 55, no. 4, pp. 482-496, April, 1967.
9. Davis, P. J.: "Interpolation and Approximation," Blaisdell Publishing Co., Waltham, Mass., 1963.
10. Hastings, C., Jr.: "Approximation for Digital Computers," Princeton University Press, Princeton, N. J., 1955.
11. Fletcher, P. E.: Optimization Techniques in System Design, in F. F. Kuo and J. F. Kaiser (eds.), "System Analysis by Digital Computer," John Wiley & Sons, Inc., New York, 1966.
12. Huelsman, L.: "Circuits, Matrices, and Linear Vector Spaces, pp. 78-91, McGraw Hill Book Company, New York, 1963.
13. Seshu, S., and M. B. Reed: "Linear Graphs and Electrical Networks," pp. 19-154, Addison-Wesley Publishing Co., Inc., Reading, Mass., 1961.
14. Guillemin, E. A.: "Theory of Linear Physical Systems," pp. 144-158. John Wiley & Sons, Inc., New York, 1963.
15. So, H. C.: Analysis and Iterative Design of Networks Using On-Line Simulation, in F. F. Kuo and J. F. Kaiser (eds.), "System Analysis by Digital Computer," pp. 34-58, John Wiley & Sons, Inc., New York, 1966.
16. Kron, G.: A Set of Principles to Interconnect the Solution of Physical Systems, *J. Appl. Phys.*, vol. 24, pp. 965-980, 1953.
17. Murray-Lasso, M. A.: The Use of the Indefinite Admittance Matrix for Computer Analysis of Circuits, *Proc. Conf. Analysis of Circuits with Digital Computer*, Nat. Univ. Mexico, Mexico City, June, 1967.
18. Daviteau, L. A.: BELNAP – A Computer Program for the AC Analysis of Linear Active and Passive Networks, unpublished work, Bell Telephone Laboratories, May, 1967.
19. *Ibid.*, BELNAP II – The Second Generation of a Computer Program for the AC Analysis of Linear Active and Passive Networks, unpublished work, Bell Telephone Laboratories, Sept., 1967.
20. Kurokawa, K.: Design Theory of Balanced Amplifiers, *Bell System Tech. J.*, vol. 44, no. 8, pp. 1675-1698, Oct., 1965.
21. Eisele, K. M., R. S. Engelbrecht, and K. Kurokawa: Balanced Transistor Amplifiers for Precise Wideband Microwave Applications, *Dig. Tech. Pap.*, Intern. Solid-State Circuits Conf., Philadelphia, pp. 18-19, Feb., 1965.

22. Murray-Lasso, M. A.: "Report on a Theoretical Investigation on Multiple Coupled Transmission Lines," unpublished work, Bell Telephone Laboratories, Sept., 1965.
23. Silverberg, M.: "Numerical Solution of Distributed Networks Containing Nonlinear Elements," doctoral dissertation, Department of Electrical Engineering, Columbia Univ., New York, 1967.
24. Cooley, J. W., and J. W. Tukey: An Algorithm for the Machine Calculation of Complex Fourier Series, *Math. of Computation*, vol. 19, pp. 297-301, April, 1965.
25. Murray-Lasso, M. A.: "Analisis y Optimizacion de Circuitos de Comunicaciones con Computadora" Memoria Segundo Congreso Panamericano de Ingenieria Mecanica, Electrica y de Ramas Afines, Caracas, Venezuela, Sept., 1967.
26. *Ibid.*, A Digital Computer Simulation of a Class of Lumped and/or Distributed Four-Ports, *Proc. 1967 ACM-SHARE Design Automation Workshop, Los Angeles*, June, 1967.
27. *Ibid.*, Unified Matrix Theory of Lumped and Distributed Directional Couplers, *Bell System Tech. J.*, vol. 47, pp. 39-71, Jan., 1968.

CAPITULO 8

Diseño óptimo mediante computadora y su aplicación a la ingeniería mecánica

Enrique Chicurel Uziel

8.1 El problema de diseño

El problema de diseño tradicional se refiere a la concepción y determinación de las características físicas de un sistema en forma tal que satisfaga básicamente los requisitos de funcionamiento cumpliendo con ciertas restricciones.

El sistema a diseñar puede ser, entre otros, una estructura, un aparato, una máquina, un sistema de control, un circuito, o bien una combinación de dos o más de los mismos.

Las restricciones se pueden referir a las limitaciones en las características mecánicas o eléctricas de los materiales, a los tamaños y capacidades de los componentes que se encuentran comercialmente, al espacio disponible, al medio ambiente, a consideraciones de seguridad y a muchas otras.

8.2 Necesidad de optimizar en diseño

Por lo general, no basta con satisfacer los requisitos de funcionamiento sino que, además, hay que satisfacerlos de acuerdo con algún criterio que depende del objeto que se persigue y la aplicación del sistema, como por ejemplo, el diseño debe ser lo más económico posible, o bien lo más eficiente, o lo más compacto, o lo más seguro. Es decir que, esencialmente, el problema de diseño es un problema de optimización, proceso que requiere una gran cantidad de operaciones aritméticas.

Cabe señalar que un diseño se mejora durante la evolución del producto, es decir, mediante el desarrollo de nuevos prototipos en donde se utilizan las experiencias logradas en la producción y aplicación de los anteriores. Este es un proceso lento y costoso pero inevitable. Sin embargo, las técnicas matemáticas de optimización y la computadora electrónica han hecho posible abreviar y abaratar dicho proceso.

Estas circunstancias, aunadas a la existencia de la fuerte competencia entre fabricantes de los países altamente industrializados ha traído como consecuencia un gran auge en el desarrollo de dichas técnicas.

Existen tres clases de optimización, a saber:

1. Optimización de magnitud
2. Optimización de forma
3. Optimización de configuración

Lo que comúnmente se entiende por optimización se refiere a la primera categoría. Sin embargo, todas son dignas de consideración.

8.3 Optimización de magnitud

Para concretar las ideas anteriores y lograr hacer optimizaciones prácticas a la mayor brevedad, echemos una ojeada a unos ejemplos de optimización de magnitud.

Ejemplo 1

Se desea diseñar una caja de cartón cilíndrica lo más económica en material posible para contener 1000 cm³ de cierto producto. Los anaqueles que se acostumbran utilizar para almacenar dichas cajas tienen una altura de 6 cm entre repisas.

Determinar las proporciones óptimas.

Solución

Formulación inicial:

$A = 2(\pi rh + \pi r^2)$	Criterio de optimización. (A, función a minimizar).
$V = \pi r^2 h$	Requisito de funcionalidad.
$h \leq h_{lim}$	Limitación.

Se especifican: V, h_{fija}

Variables: r, h
Variable restringida: h
Variable libre: r

Para obtener la formulación final, eliminamos la variable libre r

Formulación final:

$A = 2(\sqrt{\pi V}h^{1} + Vh^{-1})$ Criterio de optimización.
$h \leq h_{fija}$ Limitación.

Se especifican: V y h_{fija}
Variable: h

Recurriendo al cálculo se obtiene:

$$\frac{dA}{dh} = \sqrt{\pi V}\, h_c^{-1} - 2V h_c^{-2} = 0$$

$h_c = 1.085V^{1/3} = 10.85$ cm
puesto que $h_{fija} = 6$ cm
$h_c > h_{fija}$

Por lo tanto h_c no es la altura óptima y según podemos ver en la gráfica de la figura 1

$h_{opt} = h_{fija} = 6$ cm

y por lo tanto

$$r_{opt} = \sqrt{\frac{V}{\pi h_{opt}}} = \sqrt{\frac{1000}{\pi(6)}} = 7.25 \text{ cm}$$

$$A_{opt} = 2(\pi r_{opt} h_{opt} + \pi r^2_{opt}) =$$
$$2[\pi(7.25)(6) + \pi(7.25)^2] = 609 \text{ cm}^2$$

En la figura 1 apreciamos que cualquier punto en la región de diseño factible satisface el requisito de funcionalidad de la caja: contener 1000 cm^3. Sin embargo, sólo cuando h = 6 cm se satisface el criterio de optimización: máxima economía.

Un diseño en donde no se utilizara ningún criterio de optimización podría resultar pésimo; considérese, por ejemplo, la parte izquierda de la región de diseño factible

El método que se ilustra es debido al profesor R. C. Johnson, del Instituto Politécnico de Worcester.[13]

Ejemplo 2

Una viga de acero simplemente apoyada de 100" de largo, de sección rectangular, con 3" de peralte y 1" de ancho, soporta una carga concentrada de 720 lb en el centro.

1. Calcular σ máximo, σ = esfuerzo normal a una sección.
2. Calcular τ, máximo, τ = esfuerzo cortante transversal.

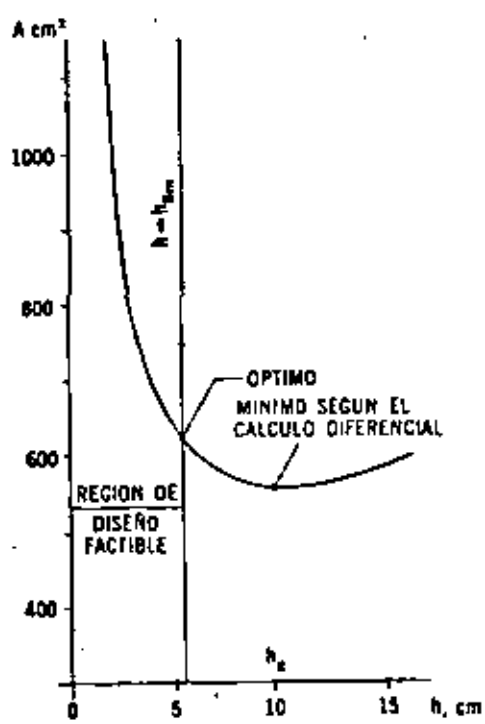


Figura 1. Optimización de caja de cartón cilíndrica.

3. Establecer una relación entre el esfuerzo cortante máximo τ en función de σ y τ_t para cualquier punto.
4. Comparar los valores numéricos obtenidos en los incisos 1 y 2 y, a la luz de ello, simplificar la relación que se estableció en el inciso 3.
5. Considerar ahora que las dimensiones de la sección de la viga no han sido determinadas. Utilizando la relación del inciso 4 determinar las dimensiones de la sección, tal que:

   El costo (peso) sea mínimo
   τ no exceda a 8000 lb/in²
   El peralte h no exceda a 3"
   La deflexión máxima no exceda 0.45"

   Proceder de la siguiente manera:

   a) Obtener la formulación inicial completa.
   b) A partir de la inicial obtener la formulación final completa.
   c) Dibujar a escala la región de diseño factible.
   d) Calcular los valores óptimos de: el área seccional (índice del costo), las dimensiones, y el esfuerzo τ.

6. Si se exige ahora que la deflexión no exceda a 0.2",

   a) Dibujar la nueva región de diseño factible.
   b) Recalcular los valores requeridos en el inciso 5d.

Solución

Nomenclatura

A = área seccional de la viga.
a = área seccional de la viga arriba (o abajo) del punto para el cual se calcula τ.
b = ancho de la viga.
c = distancia medida desde el eje neutro al punto más alejado del mismo.
Δ = deflexión máxima de la viga.
E = módulo de Young.
h = peralte de la viga.
I = momento de inercia del área seccional de la viga.
L = longitud de la viga.
M = momento flexionante.
P = carga transversal sobre la viga.
Q = primer momento del área "a" respecto al eje neutro
V = fuerza cortante.
$\bar{y}$ = distancia centroidal del área "a" medida desde el eje neutro.
$()_L$ = valor límite.

1. $\sigma_{máx} = \dfrac{M_{máx}\, c}{I} \qquad c = \dfrac{h}{2} \qquad I = \dfrac{bh^3}{12}$

$$M_{máx} = \frac{PL}{4} = \frac{(720)\ (\quad)}{4} = \quad \text{lb-in.}$$

$$\therefore \sigma_{máx} = 6\,\frac{M_{máx}}{bh^2} = \frac{6\ (18\,000}{(\quad)(3)^2} = 12\,000 \text{ lb/in}^2$$

2. $\tau_{máx} = \dfrac{V_{máx}\, Q_{máx}}{I\, b}$

$$V_{máx} = \frac{P}{2}$$

$$Q_{máx} = a\bar{y} = \left(\frac{bh}{2}\right)\frac{h}{4} = \frac{bh^2}{8}$$

$$\tau_{máx} = \frac{3}{4}\frac{P}{bh} = \frac{3}{4}\frac{(720)}{(\quad)(3)} = 180 \text{ lb/in}^2$$

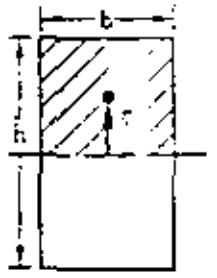


Figura 2. Ejemplo 2. Area "a" y su distancia centroidal.

3. Para cualquier punto y mediante el círculo de Mohr se obtiene

$$\tau = \sqrt{\left(\frac{\sigma}{2}\right)^2 + \tau_t^2}$$

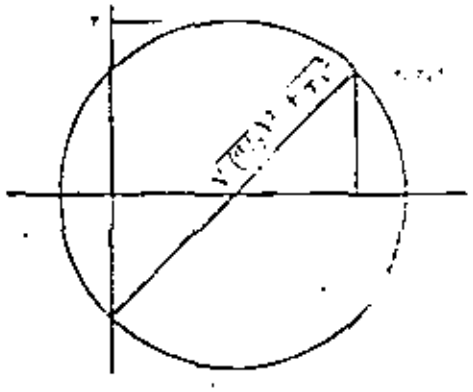


Figura 3. Ejemplo 2. Círculo de Mohr.

4. $\tau_t << \sigma$ y además

cuando $y = c$ σ es máximo y $\tau_t = 0$

cuando $y = 0$ $\sigma = 0$ y τ_t es máximo

$\therefore$ τ_t se puede considerar nulo y

$$\tau = \frac{\sigma}{2}$$

5. Consideraciones preliminares:

$$\tau = \frac{\sigma}{2} = \frac{3M}{bh^2}$$

$$\Delta = \frac{PL^3}{48EI} = \frac{ML^2}{Ebh^3}$$

Formulación inicial

$A = bh$ Criterio de optimización (A, función a minimizar)

$$\left.\begin{aligned} \tau &= 3\frac{M}{bh^2} \\ \Delta &= \frac{ML^2}{Ebh^3} \end{aligned}\right\} \text{Requisitos de funcionalidad}$$

$$\left.\begin{aligned} \tau &\leq \tau_L \\ \Delta &\leq \Delta_L \\ h &\leq h_L \end{aligned}\right\} \text{Limitaciones}$$

Se especifican: E, M, L, τ_L, Δ_L, h_L.

Variables: b, h, τ, Δ

Variables libres: b

Nótese que *las variables libres son sencillamente las que no se indican como limitadas.*

Eliminando las variables libres se obtiene la:

Formulación final

$A = \dfrac{3M}{\tau h}$ Criterio de optimización (A, función a minimizar)

$\Delta = \dfrac{L^2\tau}{3Eh}$ Requisito de funcionalidad.

$$\left.\begin{aligned} \tau &\leq \tau_L \\ \Delta &\leq \Delta_L \\ h &\leq h_L \end{aligned}\right\} \text{Limitaciones}$$

Se especifican: los parámetros E, M, L y los valores límites τ_L, Δ_L, h_L.

Variables independientes: τ, h

Variable dependiente: Δ

Observamos que, a diferencia del ejemplo 1, ahora tenemos dos variables independientes y que, por lo tanto, tendremos una región de diseño factible en dos dimensiones.

Determinemos pues las fronteras de dicha región.

Las fronteras referentes a las variables independientes quedan definidas inmediatamente por las limitaciones, de donde se obtienen sus ecuaciones:

$$\tau = \tau_L = 8\,000$$

$$h = h_L = 3$$

La frontera referente a la variable dependiente se obtiene sustituyendo la limitación correspondiente en la ecuación referente al requisito de funcionalidad

$$\tau = \frac{3E\Delta_L}{L^2}h = \frac{3(30\times10^6)(0.45)}{(100)^2}h$$

$$\tau = 4\,060\,h$$

La región de diseño factible se muestra en la figura 4.

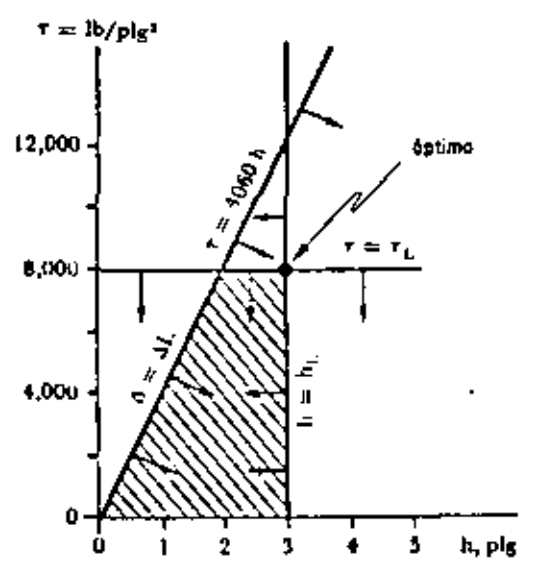


Figura 4. Ejemplo 2. Región de diseño factible para $\Delta_L = 0.45"$.

Puesto que la función objetiva es

$$A = \frac{3\,M}{\tau\,h} = \frac{54\,000}{\tau\,h}$$

A es mínimo cuando h y τ son ambas máximas dentro de la región de diseño factible o sea $h = 3''$, $\tau = 8000$ lb/in².

$$\therefore A_{\text{óptimo}} = \frac{54\,000}{3\,(8\,000)} = 2.23\ \text{in}^2$$

y el valor correspondiente de b es:

$$b = \frac{A}{h} = \frac{2.23}{3} = 0.75''$$

6. Para $\Delta_L = 0.2''$

$$\tau = 1\,800\,h$$

y la nueva región de diseño factible se muestra en la figura 5.

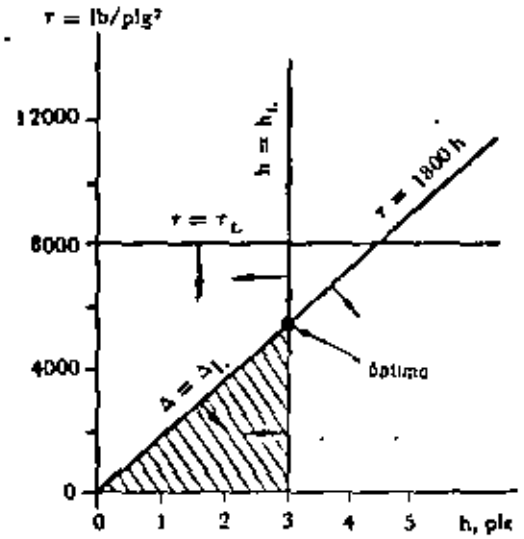


Figura 5. Ejemplo 2. Región de diseño factible para $\Delta_L = 0.2''$.

Una vez más el óptimo se obtiene de la relación:

$$A_{\text{opt}} = \frac{54\,000}{\tau_{MAX}\,h_{MAX}}$$

como se aprecia en la figura 5

$$h_{MAX} = 3''$$

y τ_{MAX} se obtiene de la intersección de las fronteras:

$$\tau = 1\,800\,h$$
$$h = h_L = 3''$$

es decir

$$\tau_{MAX} = 1\,800\,h_L = 5\,400\ \text{lb/in}^2$$

$$\therefore A_{\text{opt}} = \frac{54\,000}{(5\,400)\,(3)} = 3.33''$$

$$b_{\text{opt}} = \frac{A}{h} = \frac{3.33}{3} = 1.11''$$

Resumiendo, los valores óptimos son:

Δ_L, in	A, in²	h, in	b, in	Δ, in	τ, lb/in²
0.45″	2.23	3	0.75	0.296	8 000
0.20″	3.33	3	1.11	0.2	5 400

8.4 Consideraciones algebraicas

Si en los anteriores ejemplos se consideraron exclusivamente los requisitos de funcionalidad, se tiene un sistema con un número mayor de variables que de ecuaciones.

A continuación se resumen estas cantidades en relación a los dos primeros ejemplos.

Ejemplo	*Formulación*	*N° de variables*	*N° de ecuaciones requisito de funcionalidad*
1	Inicial	2	1
1	Final	1	0
2	Inicial	4	2
2	Final	3	1

Lo anterior quiere decir que existe un número infinito de soluciones de dicho sistema.

Si se considera ahora el conjunto de las ecuaciones referentes a los requisitos de funcionalidad y las desigualdades referentes a las limitaciones, se tiene que se han reducido los valores que pueden asumir las variables pero el número de soluciones sigue siendo infinito.

En diseño tradicional, no se define ningún criterio para seleccionar una sola de las soluciones posibles.

En cambio, en diseño óptimo sí se fija con toda precisión dicho criterio. El proceso de optimización consiste en la búsqueda de dicha solución.

Ejemplo 3

Se requiere diseñar un resorte helicoidal para un convertidor de par.[4] Ya ha sido seleccionado lo cual

tante k del resorte, así como la fuerza máxima de compresión Q.

El resorte va montado en una flecha cuyo diámetro D_f ya se fijó por lo que el diámetro interior Di queda limitado. Además por consideraciones de espacio existe un límite que no debe exceder el diámetro exterior D_e del resorte.

Ya se seleccionó el material, por lo cual ya quedaron determinados el esfuerzo permisible τ_p y el módulo de torsión G. Sin embargo, el alambre viene únicamente en M diferentes diámetros d_i.

Se desea hacer mínima la longitud cerrada del resorte (L_s cuando todas las espiras están en contacto) correspondiente a la carga máxima Q.

Las relaciones que se emplean en el diseño de resortes helicoidales son:

$$k = \frac{G d^4}{8ND_m^3}$$

$$\tau = \frac{8QD_mW}{\pi d^3}$$

$$W = \frac{4D_m - d}{4(D_m - d)} + 0.615\frac{d}{D_m}$$

d = diámetro del alambre.
D_m = diámetro medio de la espira.
N = número de espiras.
W = factor de concentración de esfuerzo de Wahl.

Establecer la formulación inicial.

A partir de la formulación inicial obtener la final.

Optimizar, dados los siguientes valores:

$k = 36$ lb/in
$Q = 415$ lb
$D_f = 2$ in
$D_{e\,max} = 4$ in
$\tau_p = 40{,}000$ lb/in^2
$G = 12 \times 10^6$ lb/in^2
$d = \frac{1}{32}, \frac{1}{16}, \frac{3}{32}, \ldots 1$ in

Solución:

Formulación inicial

$L_s = Nd$ — Criterio de optimización (L_s función a minimizar)

$$k = \frac{Gd^4}{8D_m^3N}$$

$$\tau = \frac{8QD_mW}{\pi d^3}$$

$$W = \frac{4D_m - d}{4(D_m - d)} + 0.615\frac{d}{D_m}$$

$$D_m = \frac{Di + De}{2}$$

} Requisitos de funcionalidad

$De = Di + 2d$
$Di \geq D_f$
$\tau \leq \tau_p$
$De \leq D_{e\,max}$
$d = d_1, d_2, \ldots d_M$

} limitaciones

Se especifican: k, Q, G, D_f, τ_p, $D_{e\,max}$

Variables: N, d, τ, D_m, W, Di, De
Variables libres: N, W, D_m.

Eliminando las variables libres se obtiene:

Formulación final

$$L_s = \frac{Gd^4}{8k(D_i + d)^3}$$ Criterio de optimización. L_s, función de minimizar.

$$\tau = \frac{4Q(D_i + d)}{\pi d^3}\left[\frac{4D_i + 3d}{2D_i} + 1.23\frac{d}{D_i + d}\right]$$ } Requisitos de funcionalidad

$$D_e = D_i + 2d$$

$D_i \geq D_f$
$\tau \leq \tau_p$
$D_e \leq D_{e\,max}$
$d = d_1, d_2, d_3, \ldots d_M$

} Limitaciones

Se especifican los parámetros: k, Q, G, así como los valores límite: D_f, τ_p, $D_{e\,max}$, $d_1, d_2, \ldots d_M$

Variables independientes: D_i, d
Variables dependientes: τ, D_e

Substituyendo valores numéricos obtenemos:

Formulación final

$$L_s = 3.84 \times 10^1 \frac{d^5}{(D_i + d)^3}$$

Criterio de optimización (L_s, función a optimizar)

$$\left.\begin{aligned} \tau &= 531\,\frac{(D_t + d)}{d^3}\left[\frac{4D_t + 3d}{2D_t} + \frac{1.23d}{D_t + d}\right] \\ D_c &= D_t + 2d \end{aligned}\right\} \text{Requisitos de funcionalidad}$$

$$\left.\begin{aligned} D_t &\geq 2 \\ \tau &\leq 4 \times 10^6 \\ D_c &\leq 4 \\ d &= \frac{1}{32}, \frac{1}{16}, \frac{3}{32}, \ldots 1 \end{aligned}\right\} \text{Limitaciones}$$

Variables independientes: D_t, d

Variables dependientes: τ, D_c

Determinamos primero la región de diseño factible en el sistema de coordenadas D_t, d. Se obtiene una frontera de la limitación en D_t y es la recta

$$D_t = 2$$

Nótese que por referirse a una variable independiente esta frontera es constante.

Las otras fronteras se obtienen de substituir las limitaciones de las variables dependientes en las ecuaciones referentes a los requisitos de funcionalidad.

$$4 = D_t + 2d$$

$$4 \times 10^6 = 531\,\frac{(D_t + d)}{d^3}\left(\frac{4D_t + 3d}{2D_t} + \frac{1.23d}{D_t + d}\right)$$

Como estas fronteras se refieren a las variables dependientes, no son constantes, y, en general, son curvas. Las fronteras se muestran en la figura 6. Analizando las limitaciones se concluye que la región de diseño factible es la que se muestra encerrada por las líneas sólidas.

Puesto que la función a optimizar ya no es tan sencilla, no se puede determinar el punto óptimo mediante una mera inspección de la ecuación referente al criterio de optimización.

Si no se tiene idea de optimización, lo primero que se ocurre es obtener los contornos de nivel de L_c. La ecuación se obtiene de la relación correspondiente al criterio de optimización

$$D_t = \left(3.84 \times 10^6\,\frac{d^3}{L_c}\right)^{\frac{1}{2}} - d$$

En la figura 6 se muestran cuatro contornos correspondientes a los valores de L_c = 40, 70, 150, 400 plg.

Si imaginamos a la región de diseño factible llena de contornos, llegamos a la conclusión que el óptimo

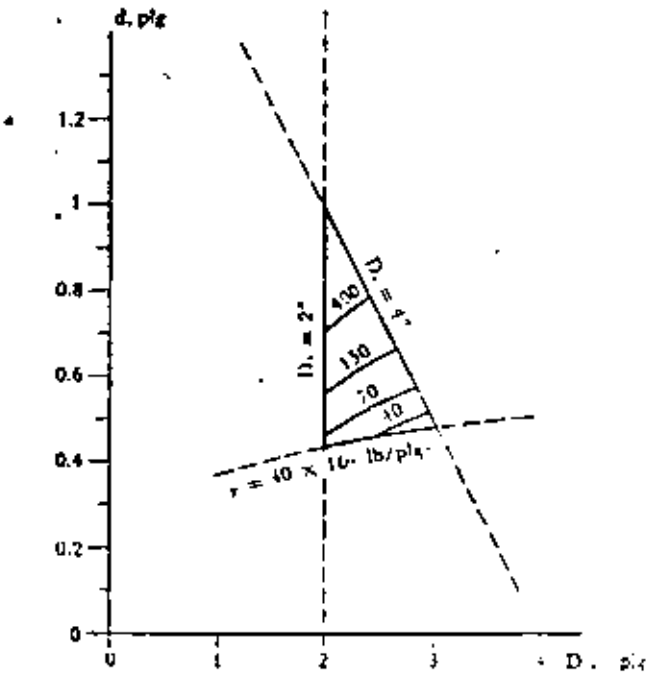


Figura 6. Ejemplo 3. Región de diseño factible y contornos de nivel de L_c.

se encuentra en el extremo derecho inferior de la región de diseño factible, es decir, d ≈ 0.48 plg.

Pero consideremos ahora a d como realmente es: una variable discreta. Esto significa que la región de diseño factible consta realmente de los segmentos de recta que se muestran en la figura 7.

Como los valores de L_c suben más lentamente a lo largo de la frontera inferior que a lo largo de la frontera derecha y como

$$\frac{15''}{32} < d < \frac{1''}{2}$$

el óptimo debe quedar en el extremo derecho del segmento inferior, por lo que las proporciones óptimas son:

$$d = 15/32''$$
$$D_t = 2.7''$$
$$D_c = 3.6''$$
$$L_c = 28''$$

calculadas a partir de la formulación final.

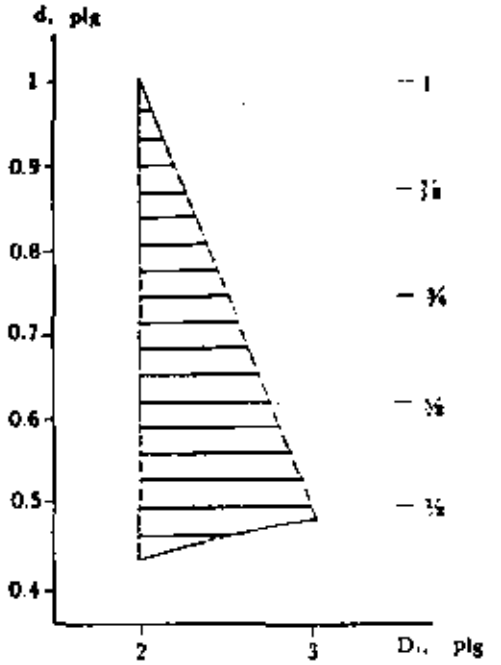


Figura 7. Ejemplo 3. Región de diseño factible tomando en cuenta los valores discretos de d.

6.5 Métodos de exploración local

La solución anterior, aunque muy ilustrativa, es sumamente ineficiente pues requiere de una gran cantidad de cómputos: por un lado, para determinar las fronteras (de hecho, la frontera inferior impuesta por la limitación en τ se calculó por tanteos); y, por otro lado, para determinar suficientes curvas de nivel para obtener una idea del comportamiento de la función a optimizar. El método sería muy difícil y laborioso si tuviéramos que lidiar con tres variables independientes y prácticamente imposible si fueran más.

Todas estas dificultades resultan del hecho que se trató de examinar la topografía de la región completa.

Hay métodos mucho más eficientes. Para entender la idea fundamental recurramos a una analogía.

Baldomero y Agripina juegan. Baldomero se halla con los ojos vendados en un punto en el interior de un predio grande, cercado, en una región montañosa y árida. Agripina le pide que, sin quitarse la venda, encuentre el punto más bajo del predio.

Baldomero se sienta en el suelo y con sus manos palpa el terreno en derredor suyo y escoge la dirección a lo largo de la cual parece descender más el terreno. Camina un trecho como de dos metros y se vuelve a sentar para palpar su derredor una vez más, repitiendo el proceso varias veces hasta encontrar un punto tal que en todas direcciones sube el terreno o bien se topa con cerca. Eufórico, le grita a Agripina que ya encontró el lugar señalado. Agripina le responde que no es cierto, pero lo conduce a otro punto inicial para darle una oportunidad más. Baldomero procede una segunda vez de idéntica manera hasta llegar a otro punto desde donde pregunta a Agripina si ése es el más bajo del predio. Agripina le responde afirmativamente y premia a Baldomero con un chocolate.

Evidentemente, en su primer intento, Baldomero llegó a un mínimo relativo. Está claro que, dadas suficientes oportunidades, Baldomero siempre podrá encontrar el mínimo absoluto, y esto, sin haber visto nunca el terreno.

Baldomero empleó un método que podríamos llamar de exploración local.

Una técnica muy útil de exploración local en optimización es el método de Box.[3]

Hemos utilizado una terminología propia de diseño pero que difiere de la utilizada por los especialistas en optimización. Así es que, primero que nada, presentamos un pequeño glosario de términos equivalentes.

Diseño	*Optimización*
Función a optimizar	= Función objetivo
Requisitos de funcionalidad	= Restricciones de igualdad
Limitaciones	= Restricciones de desigualdad
Variables independientes	= Variables de decisión
Limitaciones a las variables independientes	= Restricciones explícitas
Limitaciones a las variables dependientes	= Restricciones implícitas

Examinemos ahora el método de Box para dos variables de decisión. Supongamos concretamente que la optimización se refiere a minimización.

Se escogen cuatro puntos que satisfagan todas las restricciones, es decir, que se encuentren dentro de la región factible. Se aísla el punto más alto (el de valor más alto de la función objetivo). Se encuentra el centroide de los tres puntos restantes y se "refleja" el punto aislado a través de dicho centroide.

El punto reflejado substituye ahora al punto inicialmente aislado. De los cuatro puntos que quedan se vuelve a aislar el más alto y se repite todo el proceso anterior. De esta manera el conjunto de puntos o "Simplex" se va desplazando hacia abajo, substituyendo un punto del conjunto con cada desplazamiento. Si un punto reflejado viola una restricción explícita, se regresa a la frontera; es decir, que, a la variable que excedió su limitación, se le asigna su valor limitativo. Si un punto reflejado viola una restricción implícita, se regresa medio camino.

A GENERAL TRANSFORMATION WITH APPLICATIONS TO CIRCUIT THEORY

By

M. A. MURRAY-LASSO

Reprinted from JOURNAL OF THE FRANKLIN INSTITUTE
VOL. 285, NO. 2, FEBRUARY, 1968

Reprinted from JOURNAL OF THE FRANKLIN INSTITUTE, VOL. 285, N February, 1968
Printed in U. S. A.

A General Transformation With Applications to Circuit Theory

by M. A. MURRAY-LASSO

Bell Telephone Laboratories, Inc.
Whippany, New Jersey

ABSTRACT: *A general functional transformation is introduced and some applications to circuit theory are given. The transformation is most useful in the area of active, nonbilateral circuits. This functional transformation introduces new voltages and currents which are linear combinations of the original voltages and currents plus their derivatives, integrals or more complicated implicit integrodifferential relationships. The transformation applied to the currents may be completely unconnected to the transformation applied to the voltages.*

The transformation may be used for obtaining equivalent circuits of some driving point impedance. Another possibility is obtaining equivalent circuits of some transfer impedance but with different driving point impedances. The transformation handles with equal ease bilateral and non-bilateral circuits as well as circuits containing controlled sources. It may also handle circuits containing non-real elements. Simple numerical examples are given in active, non-bilateral, and passive synthesis, as well as in analysis.

Introduction

The transformation we introduce is most useful in the area of active, non-bilateral circuits, but may also be applied to passive, bilateral ones. The transformation is initially formulated in abstract terms, so that it can be applied to different situations. Both the dependent and independent variables are transformed, leaving, for instance, the possibility of applying one transformation to the voltages of a circuit and a completely different one to the currents. The form of the functional equation relating independent and dependent variables is left invariant. This determines the transformation of the impedance operator. After the transformation the new variables are interpreted using the same interpretation employed for the original variables.

Great liberty is allowed for the transformation operators so that when applied to the generation of equivalent circuits the transformation may go from a bilateral network to a non-bilateral one and *vice versa*. It may also leave a transfer impedance invariant while all the driving point impedances change. These problems cannot be handled with the congruent transformation which has received a good deal of attention from researchers concerned with equivalent networks through linear transformations.

Most of the applications presented in the paper are in the field of synthesis. However, to demonstrate the versatility of the functional aspect of the transformation an application to analysis is also given.

Transformation Equations

Consider the functional equation

$$y = Fx \tag{1}$$

representing the mapping by the *operator* F of the *points* (or vectors) x in the linear vector space X—called the *domain* of F—onto the *points* y in the linear vector space Y—called the *range* of F. The operator F is linear, that is, $F(\alpha x + \beta y) = \alpha Fx + \beta Fy$; α, β complex numbers.

Now make the following non-singular transformations:

$$x = P\hat{x}, \tag{2}$$

$$y = Q\hat{y}. \tag{3}$$

The points $\hat{x}$ and $\hat{y}$ and the operators P, Q in Eqs. (2) and (3) have a meaning analogous to the entities in Eq. (1). Substitution of Eqs. (2) and (3) into (1) gives $Q\hat{y} = FP\hat{x}$. Since the operation represented by Q is non-singular

$$\hat{y} = Q^{-1}FP\hat{x}, \tag{4}$$

where Q^{-1} is defined in such a way that $Q^{-1}Q\hat{x} = \hat{x}$, $QQ^{-1}x = x$, $\hat{x}$ being in the domain of Q and x being in its range.

If a new operator $\hat{F}$ is defined according to

$$\hat{F} = Q^{-1}FP, \tag{5}$$

Eq. (4) may be written

$$\hat{y} = \hat{F}\hat{x}. \tag{6}$$

The operator $\hat{F}$ given by Eq. (5) maps the points $\hat{x}$ of the space $\hat{X}$ onto the points $\hat{y}$ of the space $\hat{Y}$. The spaces to be considered in this paper are of various natures, for this reason t transformation is formulated in abstract terms borrowing the nomenclature of functional analysis. Equations (2) and (3) may be interpreted in two different ways:

a) Invariant-Object Transformations.

Consider x and $\hat{x}$ as giving the coordinates of *the same object* (point) measured in two *different frames of reference*. The operator P gives a rule for translating from one frame to the other, or

b) Invariant-Coordinate-Frame Transformations.

Consider x and $\hat{x}$ as *different objects*. The quantities x and $\hat{x}$ being the coordinates of the different points in the *same reference frame*. The operator P gives the rule for going from one point to the other. We shall consider applications of both interpretations.

A functional equation of the form of Eq. (1) which arises in circuit theory is

$$i = \mathcal{Y}v. \tag{7}$$

The voltage v and the current i are functions of the time t. The operational admittance $\mathcal{Y}$ or its inverse are often given in the form of a differential equation or in the form of the kernel of a superposition integral. The functions v and i may be considered as points in a nondenumerably infinite dimensional linear vector space and $\mathcal{Y}$ as an operator which maps the points v of the space V onto the points i of the space I. Several currents may be of interest in a circuit; hence, in Eq. (7), v and i may represent columns

$$v = \begin{bmatrix} v_1(t) \\ v_2(t) \\ \cdot \\ \cdot \\ \cdot \\ v_N(t) \end{bmatrix}, \qquad i = \begin{bmatrix} i_1(t) \\ i_2(t) \\ \cdot \\ \cdot \\ \cdot \\ i_N(t) \end{bmatrix}, \tag{8}$$

and $\mathcal{Y}$ may be specified by a set of simultaneous differential equations or as a matrix of kernels of a superposition integral. In this case v and i may still be considered as points in an infinite dimensional space and $\mathcal{Y}$ as the mapping connecting these points. With these preliminary concepts some applications of the transformation given by Eq. (4) are considered.

Application to the Synthesis of Equivalent Circuits

The circuit designer is interested in equivalent networks because some forms of circuits may be more suitable than others as regards cost, sensitivity, spread of element values, absorbing stray values, etc. Some equivalent networks are trivial variations of each other (for example, replacing a resistor by two in parallel). Others may contain elements with negative values, or non-reciprocal elements, or very different topological structures.

One way of producing equivalent networks is through linear transformations. A powerful method was proposed by Cauer (1) in 1929. Writing loop equations

n matrix form, the loop currents are transformed keeping the energy functions nvariant. This results in a congruent transformation of the loop impedance natrix. If the current in a particular loop is held invariant by suitable restrictions on the transformation matrix it follows from Lagrange's equations that he impedance of that loop must remain the same. The transformation may also e applied to a node-to-datum admittance matrix. In the Cauer transformation he elements of the transformation matrix are real constants. The transformaion received the attention of numerous investigators (2, 3, 4, 5, 8, 9, 12). Cauer showed that the positive definite or semidefinite character of the paramter matrices is both necessary and sufficient for realizability with positive ircuit elements as long as ideal transformers are allowed. This property is preerved by a congruent transformation. This implies that if the node-to-datum dmittance matrix of a passive network is Cauer transformed, the resultant network will be passive as far as the node-to-datum ports are concerned, even hough some elements may be active. However, it is clear that given two circuits, ne of which is passive and the other active, as seen from a set of node-to-datum orts, both may have the same driving point impedance at one port. But the Cauer transformation cannot produce the active circuit from the passive one. Furthermore, with the Cauer transformation the final circuits will be bilateral f the original circuit was bilateral. This is a consequence of the fact that a congruent transformation does not destroy the symmetry of a matrix. For these reasons, in synthesis in which passive, active, bilateral and nonbilateral networks are of interest congruent transformations have restricted applicability.

In this paper the circuit elements assumed to be at the disposal of the designer are: constant resistors, inductors, capacitors (all both positive and negative), gyrators and controlled sources. For these elements the complex frequency domain representation is convenient. However if the transformation is applied o time varying circuits it might be more convenient to work in the time domain.

The attitude of allowing for greater generality is further motivated by the act that with the recent developments in monolithic and other integrated circuits, more than before, the circuit designer is interested in equivalent circuits ontaining active non-bilateral elements. The microminiature transistor technology has advanced to the point that often it is more desirable to introduce ra ors than passive elements. This situation implies that methods which a ew years ago would not have received the attention of circuit designers have become relevant given the present state of the art.

Transformation for Invariant Driving Point Impedance

Suppose a circuit is analyzed on the node basis using ground as the datum ode. The equilibrium equations may be written symbolically in the time domain s in Eq. (7); v and i are the vectors given by Eqs. (8) and $\mathcal{Y}$ represents the differential equations linking v and i. The complex frequency domain representation f the circuit is already an application of Eq. (4), if the variables of Eqs. (2),

(3) and (4) are identified in the following way:

$$x = v(t), \quad \dot{x} = V(s), \quad P = E\int_{-\infty}^{\infty} [\cdot]e^{-st}\,dt, \tag{9}$$

$$y = i(t), \quad \dot{y} = I(s), \quad Q = E\int_{-\infty}^{\infty} [\cdot]e^{-st}\,dt. \tag{10}$$

(In the definition of the integral operators P and Q, E stands for the $N \times N$ unit matrix.) Equation (6) reads, for this case

$$I(s) = Y(s)V(s), \tag{11}$$

where

$$I(s) = \begin{bmatrix} I_1(s) \\ I_2(s) \\ \cdot \\ \cdot \\ \cdot \\ I_N(s) \end{bmatrix}, \quad V(s) = \begin{bmatrix} V_1(s) \\ V_2(s) \\ \cdot \\ \cdot \\ \cdot \\ V_N(s) \end{bmatrix}, \tag{12}$$

are the double-sided Laplace transforms of the current and voltage vectors; $Y(s)$ is an $N \times N$ matrix whose entries are, for RLC circuits, ratios of polynomials of the complex variable s of the Laplace transformation.

Equation (11) is an example of the invariant-object interpretation of a transformation, since the new variables represent the same currents and the same voltages in a different frame of reference known as the complex frequency domain. We assume that this transformation has been done and Eq. (11) is the starting point. No transformers or mutual inductances are considered to be present in the circuits discussed. In Eq. (12) V_j is the voltage of the jth node with respect to the datum; I_j is the sum of the currents injected into node j by the current sources; Y_{jj} is the sum of all the admittances connected to node j; Y_{jk} is the negative of the admittance connected between nodes j and k.

Consider a network with only one current source connected to the two terminals forming a port whose driving point impedance is of interest. Make the node into which the current is entering from the source node 1, and the other the datum. Hence,

$$I(s) = \begin{bmatrix} I_1(s) \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}, \tag{12'}$$

Now subject the voltage and current vectors to non-singular transformations, defined by

$$I(s) = AÎ(s), \tag{13}$$

$$V(s) = BV̂(s). \tag{14}$$

Let the operators A and B be $N \times N$ matrices whose entries are ratios of polynomials of s. If the form of the functional equation is kept invariant, Eq. (11) is transformed into

$$Î = ŶV̂, \tag{15}$$

where

$$Ŷ = A^{-1}YB. \tag{16}$$

The transformation given by Eqs. (13) and (14) is an invariant-coordinate-frame transformation, that is, the new current and voltage vectors will be different physically. They will be measured in the same reference frame as the old currents and voltages, which means that after the transformation the interpretation of $Î$, $V̂$, $Ŷ$ is based on the same criterion that was used for the interpretation of I, V, Y.

If it is desired to keep the driving point impedance between node 1 and the datum invariant, the operators A and B must be restricted somewhat. For the original circuit, the driving point impedance from node 1 to ground is

$$z = V_1/I_1. \tag{17}$$

In the new circuit the driving point impedance from node 1 to ground is

$$z = V̂_1/Î_1. \tag{18}$$

If Eqs. (18) and (17) are equated, the operators are to be restricted in such a way that

$$V_1/I_1 = V̂_1/Î_1 \tag{19}$$

is satisfied. Hence, a possibility is to choose A and B such that

$$V̂_1 = βV_1, \tag{20}$$

$$Î_1 = βI_1, \tag{21}$$

where β is an arbitrary ratio of polynomials of s. If the matrix B is of the form

$$B = \begin{bmatrix} 1/\beta & 0 & \cdots & 0 \\ B_{21} & B_{22} & \cdots & B_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ B_{N1} & B_{N2} & \cdots & B_{NN} \end{bmatrix}, \tag{22}$$

and the matrix $A^{-1} = C$ is of the form

$$A^{-1} = C = \begin{bmatrix} \beta & C_{12} & \cdots & C_{1N} \\ C_{21} & C_{22} & \cdots & C_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ C_{N1} & C_{N2} & \cdots & C_{NN} \end{bmatrix}, \tag{23}$$

Eq. (19) will be satisfied and the transformed circuit will have the same driving point impedance between node 1 and the datum as the original circuit. The transformed circuit will, in general, contain controlled sources. The short-circuit admittance matrix $Ŷ$ of the new circuit is given by Eq. (16). For further clarification at this point we introduce a simple numerical example.

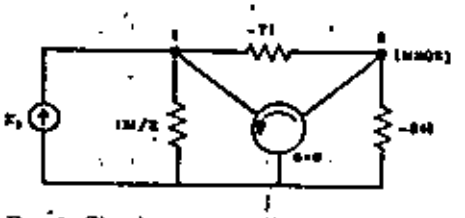


FIG. 1. Circuit corresponding to Y of Eq. 24.

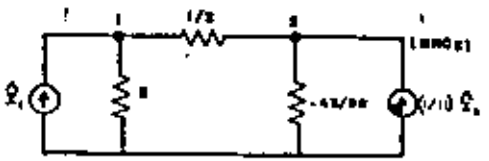


FIG. 2. Circuit corresponding to $Ŷ$ of Eq. 30 and $Î$ of Eq. 31.

Consider the circuit of Fig. 1. The driving point impedance from node 1 to the datum is one ohm. The short-circuit node-to-datum admittance matrix is

$$Y = \begin{bmatrix} -11/2 & 71 \\ 71 & -770 \end{bmatrix} + \begin{bmatrix} 0 & -6 \\ 6 & 0 \end{bmatrix} = \begin{bmatrix} -11/2 & 65 \\ 77 & -770 \end{bmatrix}. \tag{24}$$

By applying the transformation of Eq. (16) it is desired to obtain a *bilateral* circuit that has the same driving point impedance from node 1 to the datum. Using Eqs. (16), (22) and (23) yield

$$Ŷ = \begin{bmatrix} \beta & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} -11/2 & 65 \\ 77 & -770 \end{bmatrix} \begin{bmatrix} 1/\beta & 0 \\ B_{21} & B_{22} \end{bmatrix}. \tag{25}$$

Performing the multiplications indicated

$$Ŷ = \begin{bmatrix} -\tfrac{11}{2} + 77(C_{12}/\beta) + (65\beta - 770C_{12})B_{21} & (65\beta - 770C_{12})B_{22} \\ (-\tfrac{11}{2}C_{21} + 77C_{22})(1/\beta) + (65C_{21} - 770C_{22})B_{21} & (65C_{21} - 770C_{22})B_{22} \end{bmatrix}. \tag{26}$$

Since it is desired to obtain a bilateral circuit, the matrix $Ŷ$ must be symmetric.

This imposes the following constraint:

$$(-\tfrac{1}{2}C_{11} + 77C_{12})(1/\beta) + (65C_{11} - 770C_{12})B_{11} = (65\beta - 770C_{13})B_{21}, \quad (27)$$

from which one of the arbitrary parameters β, C_{13}, C_{21}, C_{22}, B_{21}, B_{22} may be eliminated. All the rest of the parameters remain arbitrary. The following choice

$$\beta = 1, \quad C_{13} = \tfrac{1}{7}, \quad C_{21} = \tfrac{1}{77}, \quad C_{22} = 0, \quad B_{21} = 0, \quad (28)$$

gives, from Eq. (27)

$$B_{22} = \tfrac{1}{198}, \quad (29)$$

and yields the following $\hat{Y}$ matrix from Eq. (26):

$$\hat{Y} = \begin{bmatrix} 11/2 & -1/2 \\ -1/2 & 13/198 \end{bmatrix}. \quad (30)$$

According to the choice made

$$\begin{bmatrix} \hat{I}_1 \\ \hat{I}_2 \end{bmatrix} = \begin{bmatrix} 1 & 1/7 \\ 1/11 & 0 \end{bmatrix} \begin{bmatrix} I_1 \\ 0 \end{bmatrix}.$$

That is,

$$\hat{I}_1 = I_1, \qquad \hat{I}_2 = \tfrac{1}{77} I_1 = \tfrac{1}{77} \hat{I}_1. \quad (31)$$

Hence, a controlled source of the value shown by Eq. (31) must be connected from node 2 to the datum. The circuit realizing $\hat{Y}$ of Eq. (30) and with the controlled current source indicated by Eq. (31) appears in Fig. 2. Analysis of the circuit of Fig. 2 shows that the driving point impedance from node 1 to the datum is one ohm, the same as that of Fig. 1. With this transformation the gyrator was eliminated at the expense of having a controlled source. In the next section we show how to avoid the controlled sources.

Invariant Driving Point Impedance Without Introducing Controlled Sources

In the circuit of Fig. 2 a controlled current source appeared due to the fact that the matrix A^{-1} has non-zero elements in the first column. If the matrix C of Eq. (22) is restricted to be of the form

$$A^{-1} = C = \begin{bmatrix} \beta & C_{12} & \cdots & C_{1N} \\ 0 & C_{22} & \cdots & C_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & C_{N2} & \cdots & C_{NN} \end{bmatrix}. \quad (32)$$

No controlled sources appear in the new circuit. The rest of the procedure is identical to the previous one. Returning to Eq. (25), if $C_{11} = 0$ so that Eq. (32) be satisfied, and the following choice is made

$$\beta = 77, \quad C_{12} = 13, \quad C_{13} = -13, \quad B_{21} = \tfrac{5}{5005}, \quad B_{22} = \tfrac{5}{5005},$$

the resulting $\hat{Y}$ is

$$\hat{Y} = \begin{bmatrix} 3/2 & -1 \\ -1 & 2 \end{bmatrix}.$$

The corresponding circuit is shown in Fig. 3, where the driving point impedance from node 1 to the datum is unity, as in Figs. 1 and 2. However, no gyrators or controlled sources appear. It is possible also to go from a transformed circuit to the original one by applying another transformation. From Eq. (16), solving for Y, $Y = A\hat{Y}B^{-1}$, which expresses the same transformation as Eq. (16) since A and A^{-1} are of the same form and likewise B and B^{-1}. The fact is obvious if A^{-1} is of the form of Eq. (23) since all the elements are arbitrary. For matrices of the form given by Eq. (22) note that, except the first, all cofactors of the elements in the first column of the transpose of B are zero. Hence, B^{-1} will have zeros in the elements of the first row (except the first element). A similar statement can be made for matrices of the form given in Eq. (32). This means that we may start with a bilateral circuit and end with a non-bilateral circuit and *vice versa*.

Invariant Transfer Impedance

Instead of keeping a driving point impedance invariant it may be desired to keep a transfer impedance invariant, say, the transfer impedance between the ports formed of node 1 and ground and node 2 and ground. Then, following a procedure similar to the one followed in Eqs. (17) through (23), it is sufficient to restrict B and C to be of the form

$$B = \begin{bmatrix} B_{11} & B_{12} & B_{13} & \cdots & B_{1N} \\ 0 & 1/\beta & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ B_{N1} & B_{N2} & B_{N3} & \cdots & B_{NN} \end{bmatrix}, \quad (33)$$

$$C = A^{-1} = \begin{bmatrix} \beta & C_{12} & \cdots & C_{1N} \\ C_{21} & C_{22} & \cdots & C_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ C_{N1} & C_{N2} & \cdots & C_{NN} \end{bmatrix}. \quad (34)$$

This gives a circuit in which z_{21} is kept invariant and controlled sources appear. If no controlled sources are desired, the matrix C should be of the form of Eq. (32)

so that the currents entering nodes 2, 3, $\cdots$, N are not related to the current entering node 1, as indicated by Eq. (13). If we desire to keep z_{11} invariant, rather than starting with a circuit in which $I(s)$ is of the form of (12') we should have

$$I(s) = \begin{bmatrix} 0 \\ I_1(s) \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}. \tag{35}$$

The matrix B should be of the form of B of Eq. (22) and $C = A^{-1}$ should be of the form

$$C = A^{-1} = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1N} \\ C_{21} & \beta & \cdots & C_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ C_{N1} & C_{N2} & \cdots & C_{NN} \end{bmatrix}. \tag{36}$$

The transformed circuit will have controlled sources. If no controlled sources are desired, C should be of the form

$$C = \begin{bmatrix} C_{11} & 0 & \cdots & C_{1N} \\ C_{21} & \beta & \cdots & C_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ C_{N1} & 0 & \cdots & C_{NN} \end{bmatrix}. \tag{37}$$

Note that Caver's transformation theory cannot handle this problem (10). It would be obvious how to extend these concepts to maintain invariant several driving point and/or transfer impedances.

Impedance Driving Point or Transfer Impedance to Yield a Bilateral Circuit

If the original Y is symmetric and

$$A^{-1} = B^T, \tag{38}$$

where the superscript T denotes transposition; then according to Eq. (16)

$$\hat{Y} = B^T Y B = \hat{Y}^T. \tag{ }$$

Hence $\hat{Y}$ is symmetric. Equation (38) is a *sufficient* condition for symmetry of

$\hat{Y}$ if Y is symmetric. If Eq. (39) is used, β of Eqs. (20) and (21) is forced to be unity.

$$\hat{Y} = \begin{bmatrix} 1 & B_{21} & \cdots & B_{N1} \\ 0 & B_{22} & \cdots & B_{N2} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & B_{2N} & \cdots & B_{NN} \end{bmatrix} \begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1N} \\ Y_{12} & Y_{22} & \cdots & Y_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ Y_{1N} & Y_{2N} & \cdots & Y_{NN} \end{bmatrix} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ B_{21} & B_{22} & \cdots & B_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ B_{N1} & B_{N2} & \cdots & B_{NN} \end{bmatrix}. \tag{40}$$

So far, the entires B_{jk} are still arbitrary rational functions of s. If they are restricted to be constants the transformation reduces to the Cauer transformation. Although condition (38) is sufficient for ending with a symmetric $\hat{Y}$, if the original Y is symmetric, it is not necessary (7). All that is necessary is that

$$\hat{Y} = \hat{Y}^T; \tag{40'}$$

this gives, using Eq. (16),

$$A^{-1} Y B = B^T Y^T (A^{-1})^T. \tag{41}$$

Equation (41) represents $(N^2 - N)/2$ equations with which we may eliminate as many variables from the matrices B and A^{-1}, leaving the rest of the entries arbitrary. Equation (41) is especially useful if the original Y is not symmetric, and when keeping transfer impedances invariant while changing all the driving point impedances. Its use is illustrated for the example of Fig. 1. For that case, the matrix Eq. (41) reduces to the single scalar Eq. (27).

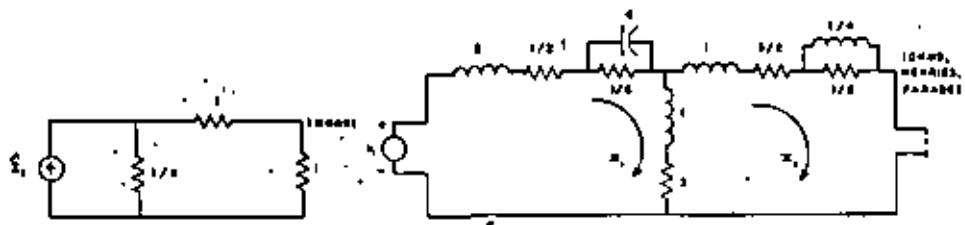


FIG. 3. Circuit equivalent to those of Figs. 1 and 2.

FIG. 4. Circuit to be transformed using the loop basis.

Passive Transformless Synthesis by Transformation

If the driving point impedance of interest is realizable and we wish to obtain an RLC circuit (no transformers or mutual inductances) with positive elements, besides wanting $\hat{Y}$ to be symmetric, it is sufficient that the negative of the terms off the main diagonal as well as the sum of the entries of each row are positive real functions. Few general results exist to determine A^{-1} and B which guarantees that $\hat{Y}$ will satisfy these conditions.

Pantell (19), Guillemin (13), Duda (5), Darlington (12), Schneider (17), and Cederbaum (16) have some results for two element [illegible]works. Schoeffler

(15) has done work in the RLC problem. However, the problem is far from solved. Newcomb (18) treats the problem allowing the presence of transformers.

Invariant Driving Point Impedance by Transforming Circuits on the Loop Basis

What has been said for the nodes basis can be extended to the loop basis. The loop basis is appropriate for keeping short-circuit driving point and transfer admittances invariant on ports formed by making plier-type of entries into the loops. (In the node basis soldering-iron-type of entries at node pairs were made.) As an example of a transformation on the loop basis, we consider a simple problem in passive synthesis. This example illustrates the use of transformation matrices whose entries are functions of s, thus exploiting the functional character of the transformation. Consider the circuit of Fig. 4 whose loop impedance matrix is

$$Z = \left[\begin{array}{c|c} 3s + 3 + \dfrac{s+2}{2s+3} & -(s+3) \\ \hline -(s+3) & s + 3 + \dfrac{(s+1)(s+3)}{s+2} \end{array}\right].$$

The matrix Z may be transformed into a new matrix $\hat{Z}$ according to

$$\hat{Z} = CZB. \tag{42}$$

In order to keep the short-circuit driving point admittance of a plier entry at loop 1 invariant, C and B will be of the forms

$$A^{-1} = C = \begin{bmatrix} \beta & p \\ 0 & r \end{bmatrix}, \qquad B = \begin{bmatrix} 1/\beta & 0 \\ m & n \end{bmatrix}. \tag{43}$$

Since C and B are of the forms indicated by Eqs. (22) and (32) the new circuit will not contain controlled sources. If, besides, Eq. (41) is satisfied the resulting circuit will be bilateral. If the multiplications indicated by Eq. (42) are performed with the aid of Eq. (43), an equation similar to Eq. (27) is written and m eliminated. The new matrix $\hat{Z}$ may be written as follows:

$$\hat{Z} = \left[\begin{array}{c|c} Z_{11} + \dfrac{np^2}{r} Z_{22} + Z_{12}\left[\dfrac{2np}{\beta r} + \dfrac{Z_{12}}{Z_{22}}\left(\dfrac{n}{\beta^2 r} - 1\right)\right] & \dfrac{nZ_{12}}{\beta} + npZ_{22} \\ \hline \dfrac{nZ_{12}}{\beta} + npZ_{22} & nrZ_{22} \end{array}\right]. \tag{44}$$

In Eq. (44) the elements β, p, r, n are still arbitrary and may be manipulated to make the final circuit realizable.

The following choice in the parameters in Eq. (43)

$$\beta = 1, \quad p = 0, \quad r = 1, \quad n = (s+2)/(s+3),$$

and

$$m = s + 2/(2s+3)(s+3),$$

gives the new loop impedance matrix

$$\hat{Z} = \begin{bmatrix} 3s+3 & -(s+2) \\ -(s+2) & 2s+3 \end{bmatrix},$$

whose realization is shown in Fig. 5. The short-circuit admittance at loop 1 of both circuits of Figs. 4 and 5 is

$$y = (2s+3)/(5s^2 + 11s + 5).$$

The transformation eliminates the capacitance in the circuit which is redundant. We emphasize that in general it is very difficult to choose the transformation coefficients such that the resulting network is realizable with positive elements. Furthermore, with respect to transformations in the loop basis it is well known that circuits with more than three independent meshes must meet complicated constraints unless transformers or some other non-conductive couplings are employed (14).

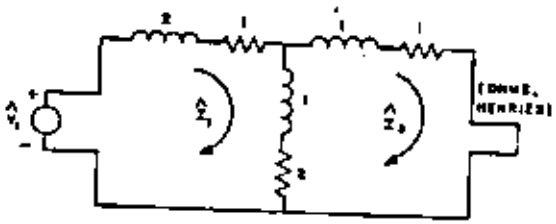


FIG. 5. Circuit equivalent to that of Fig. 4.

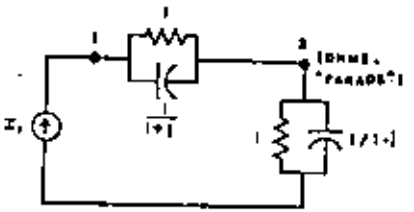


FIG. 6. "Foster Realization" of a circuit with complex poles.

Non-real Transformations

In the previous example a realizable circuit was transformed into a realizable one. It is possible also to transform a realizable circuit into a non-realizable one and *vice versa*. An extreme case is provided by the following example:

The impedance

$$z = \frac{2s+4}{s^2 + 2s + 2} = \frac{1+j}{s+1+j} + \frac{1-j}{s+1-j},$$

can be realized in a Foster form, see Fig. 6. The Y of the circuit of Fig. 6 is

$$Y = \left[\begin{array}{c|c} 1 + \frac{s}{1+j} & -\left(1 + \frac{s}{1+j}\right) \\ \hline -\left(1 + \frac{s}{1+j}\right) & \left(\frac{1}{1+j} + \frac{1}{1-j}\right)s + 2 \end{array}\right].$$

Using Eq. (44) (interpreted for a $\hat{Y}$ matrix instead of a $\hat{Z}$ matrix) and choosing $\beta = 1, p = 0, r = 1, n = 0$,

$$\hat{Y} = \left[\begin{array}{c|c} (s/2) + 1/(s+2) & \begin{matrix}0\\-\\0\end{matrix} \\ \hline 0 & \end{array}\right],$$

w realization is shown in Fig. 7. Analysis shows that

$$t = \frac{2s + 4}{s^2 + 2s + 2}.$$

Altering the Driving Point Impedance Through Transformations

The driving point impedance need not remain invariant after the transfor- , it can be scaled (6) by a factor α, where α is a ratio of polynomials. For this transformation the operator B of Eq. (22) is of the form

$$B = \begin{bmatrix} 1/\alpha\beta & 0 & \cdots & 0 \\ B_{21} & B_{22} & \cdots & B_{2N} \\ B_{N1} & B_{N2} & \cdots & B_{NN} \end{bmatrix}, \tag{45}$$

Hence, if the driving point impedance of the original circuit is z, the one of the transformed circuit will be αz.

Changing the Order of the Matrices

With the transformations described so far the order of the final admittance matrix is the same as the one of the original matrix. Although it is possible to ntroduce new nodes, the added nodes will not be connected to all the others. f it is desired to increase the complexity of the networks it is possible to do so by introducing a larger matrix Y_A partitioned in the following diagonal form:

$$Y_A = \left[\begin{array}{c|c} Y & 0 \\ \hline 0 & \Gamma \end{array}\right],$$

where Y is the original $N \times N$ admittance matrix, 0 is the zero matrix, and Γ is an $M \times M$ matrix. The matrix Y_A, which is now $(N + M) \times (N + M)$ may be transformed with transformation matrices of the same order. (See 12).

Application of the Transformation to Analysis

So far the transformation has been applied to the synthesis of equivalent circuits. An application to analysis is now considered which illustrates the advantage of having formulated the transformation in abstract terms. The transformation is of the *invariant-object* type.

In power systems one deals with a single frequency, steady state problem. Kenelly introduced in 1893 the idea of handling the problem with complex numbers. An alternate treatment using only real matrices is possible (11).

Define the operator L as follows:

Domain of L: all 2-vectors with real components;
Range of L: all sinusoids of arbitrary phase and fixed frequency ω;

$$L\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = V_1 \sin \omega t + V_2 \cos \omega t. \tag{46}$$

The operator L has an inverse L^{-1} whose domain is the range of L and whose range is the domain of L and

$$L^{-1}(V_1 \sin \omega t + V_2 \cos \omega t) = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}. \tag{47}$$

It can be shown that if V_a and V_b are 2-vectors L satisfies

$$L(\alpha V_a + \beta V_b) = \alpha L V_a + \beta L V_b.$$

Let

$$y = (d/dt)x \tag{48}$$

be transformed according to

$$y = L\hat{y}, \tag{49}$$

$$x = L\hat{x}. \tag{50}$$

Hence,

$$\hat{y} = L^{-1}(d/dt)L\hat{x}. \tag{51}$$

Now let x and y be sinusoids given by

$$x = X_1 \sin \omega t + X_2 \cos \omega t, \tag{52}$$

Using Eqs. (49) and (50)

$$\dot{x} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, \quad (54)$$

$$\dot{y} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}. \quad (55)$$

Substitution of (54) and (55) gives

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = L^{-1}\frac{d}{dt}L\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}. \quad (56)$$

Using definition Eq. (46)

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = L^{-1}(d/dt)\,(X_1 \sin \omega t + X_2 \cos \omega t)$$

$$= L^{-1}(-\omega X_1 \cos \omega t + \omega X_2 \sin \omega t). \quad (57)$$

Using Eq. (47) in Eq. (57)

$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} \omega X_2 \\ -\omega X_1 \end{bmatrix} = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix}\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \quad (58)$$

Hence comparing (58) and (51)

$$L^{-1}(d/dt)L = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix}. \quad (59)$$

In a similar manner it can be shown that for

$$F_1 = \int [\cdot]\,dt$$

$$L^{-1}F_1L = \begin{bmatrix} 0 & -1/\omega \\ 1/\omega & 0 \end{bmatrix} \quad (61)$$

and for

$$F_2 = K[\cdot] \quad (62)$$

(K a multiplicative constant)

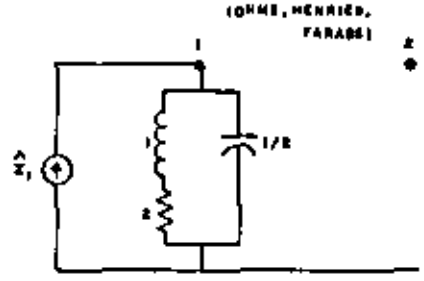


FIG. 7. Circuit equivalent to that of Fig. 6.

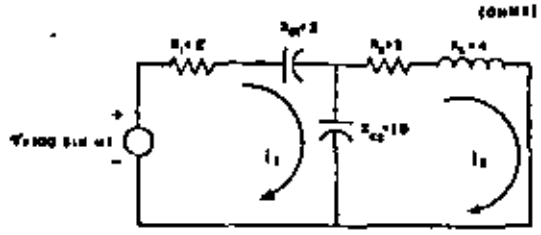


FIG. 8. Circuit to be analyzed without using complex numbers.

$$L^{-1}F_2L = \begin{bmatrix} K & 0 \\ 0 & K \end{bmatrix}. \quad (63)$$

If x represents current and y represents voltage then a series RLC circuit of operational impedance

$$z = R + L(d/dt) + C^{-1}\int dt$$

in the time domain has an operational impedance in the new coordinate system of

$$\mathbf{z} = \begin{bmatrix} R & 0 \\ 0 & R \end{bmatrix} + \begin{bmatrix} 0 & \omega L \\ -\omega L & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1/\omega C \\ 1/\omega C & 0 \end{bmatrix} \quad (64)$$

$$\mathbf{z} = \begin{bmatrix} R & \omega L - 1/\omega C \\ 1/\omega C - \omega L & R \end{bmatrix} = \begin{bmatrix} R & X \\ -X & R \end{bmatrix} \quad (65)$$

where $X = \omega L - 1/\omega C$. For a multiloop system, such as that shown in Fig. 8, transforming separately each v_k, z_{kj}, and i_j in

$$\begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix} = \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix}\begin{bmatrix} i_1(t) \\ i_2(t) \end{bmatrix} \quad (66)$$

and writing the resultant equation in partitioned form, the following equation is obtained in the new coordinate system:

$$\begin{bmatrix} [\hat{V}_1]_1 \\ [\hat{V}_2]_1 \\ [\hat{V}_1]_2 \\ [\hat{V}_2]_2 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} R & X \\ -X & R \end{bmatrix}_{11} & \begin{bmatrix} R & X \\ -X & R \end{bmatrix}_{12} \\ \begin{bmatrix} R & X \\ -X & R \end{bmatrix}_{21} & \begin{bmatrix} R & X \\ -X & R \end{bmatrix}_{22} \end{bmatrix}\begin{bmatrix} [\hat{I}_1]_1 \\ [\hat{I}_2]_1 \\ [\hat{I}_1]_2 \\ [\hat{I}_2]_2 \end{bmatrix} \quad (67)$$

which may be written, using the values of Fig. 8

$$\begin{bmatrix}\begin{bmatrix}100\\0\end{bmatrix}\\\begin{bmatrix}0\\0\end{bmatrix}\end{bmatrix} = \begin{bmatrix}\begin{bmatrix}2 & -12\\12 & 2\end{bmatrix} & \begin{bmatrix}0 & 10\\-10 & 0\end{bmatrix}\\\begin{bmatrix}0 & 10\\-10 & 0\end{bmatrix} & \begin{bmatrix}3 & -6\\6 & 3\end{bmatrix}\end{bmatrix}\begin{bmatrix}\begin{bmatrix}I_1\\I_2\end{bmatrix}_1\\\begin{bmatrix}I_1\\I_2\end{bmatrix}_2\end{bmatrix} \tag{68}$$

Solving for the currents,

$$I = \begin{bmatrix}\begin{bmatrix}11.75\\1.73\end{bmatrix}\\\begin{bmatrix}13.9\\9.85\end{bmatrix}\end{bmatrix}.$$

Using Eq. (49) this implies that in Fig. 8

$$i_1 = 11.75 \sin \omega t + 1.73 \cos \omega t$$
$$i_2 = 13.9 \sin \omega t + 9.85 \cos \omega t.$$

Here, the use of complex numbers is avoided. This method is useful for digital computer calculations in which a FORTRAN subroutine for inverting real matrices in double precision is available. In this case the subroutine as it stood ld not handle double precision complex matrices.

Conclusions

The paper has introduced a functional transformation of a very general character and illustrated some of the applications to circuit analysis and synt . Alternatively, one may apply the transformation to A, B, C, D paramatrices or other parameter representations of circuits. Other possible applications include distributed circuits and time varying circuits. This author l that transformation theory applied to circuit theory is a wide open field of rch that has thus far remained largely untapped.

Ackn ledgment

The author is indebted to Prof. P. Moon and Prof. E. A. Guillemin for helpful discussions.

References

(1) W. Cauer, "Vierpole," *Elek. Nachr. Tech.*, Vol. 6, pp. 272–282, 1929.
(2) N. Howitt, "Equivalent Electi Networks," *Proc. IRE*, Vol. 20, pp. 1042–1051, 1932.
(3) R. S. Burlington, "*R*-Matri Equivalent Networks," J . *Math. Phys.*, Vol. 16, pp. 85–103, 1937.

(4) E. A. Guillemin, "Transformation Theory Applied to Linear Active and/or Non-bilateral Networks," *IRE Trans. on Circ. Thy.*, pp. 106–111, Sept. 1957.
(5) R. Duda, "Equivalent and Optimal Equivalent Electrical Networks," Ph.D. Diss., M.I.T., 1962.
(6) M. A. Murray-Lasso, "Generalized Impedance Leveling in Network Synthesis," *Proc. 2nd Annual Allerton Conf. on Circ. and Syst. Thy.*, pp. 820–840, 1964.
(7) M. A. Murray-Lasso, "Matrix Transformations in Circuit Theory," M.S. Thesis M.I.T., 1962.
(8) L. P. Huelsman, "Circuits, Matrices, and Linear Vector Spaces," New York, McGraw-Hill, 1963.
(9) E. Guillemin, "Synthesis of Passive Networks," New York, John Wiley, 1957.
(10) S. Darlington, "A Survey of Network Realisation Techniques," *IRE Trans. on Circ. Thy.*, Vol. CT-2, 1955.
(11) P. Moon and D. E. Spencer, "A New Mathematical Representation of Alternating Currents," *J. Tensor Soc. of Japan*, 1964.
(12) S. Darlington, "On Three Terminal Circuits Without Transformers," *Proc. Third Annual Allerton Conf. on Circ. and Syst. Thy.*, Univ. of Ill., Oct. 1965.
(13) E. Guillemin, "Theory of Linear Physical Systems," New York, John Wiley, 1963.
(14) R. M. Foster, "Topologic and Algebraic Considerations in Network Synthesis," *Proc. Symp. on Modern Network Synthesis*, Brooklyn Polytech. Inst., 1952.
(15) D. A. Calahan, "Modern Network Synthesis," Vol. 2, New York, Hayden Book Co., 1964.
(16) I. Cederbaum, "Dominant Matrices and Their Application to Network Synthesis Under Topological Constraints," *Franklin Inst.*, Dec. 1964.
(17) A. J. Schneider, "RC Driving-Point Impedance Realization by Linear Transformations," *IEEE Trans. on Circ. Thy.*, Vol. CI-13, No. 3, pp. 265–271, Sept. 1966.
(18) R. W. Newcomb, "Linear Multiport Synthesis," New York, McGraw-Hill, 1966.
(19) R. E. Pantell, "New Methods of Driving-Point and Transfer Function Synthesis," Tech. Rept. No. 76, Electronics Res. Lab. Stanford Univ., Stanford, Calif., July 19, 1954.

ANALISIS DE CIRCUITOS CON COMPUTADORA DIGITAL

M A MURRAY LASSO
L P MC NAMEE

281

ENERO 1971

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Análisis de circuitos con computadora digital

M. A. Murray Lasso*
L. P. McNamee**

RESUMEN

Se presenta una vista panorámica con apuntes bibliográficos sobre diseño de circuitos con computadora digital. Se ilustran sus posibilidades con un ejemplo específico de un circuito electrónico que se analiza con un programa general.

ABSTRACT

The field of computer-aided circuit design is briefly surveyed and a bibliography is given. Its posibilities are illustrated with a specific example by analizing an electronic circuit with a general purpose program.

1. INTRODUCCION

La práctica del análisis de circuitos con computadora digital es tan vieja como la computadora misma. Inmediatamente después de la Segunda Guerra Mundial se comenzaron a analizar filtros en el dominio de la frecuencia en una computadora de relevadores en los Laboratorios *Bell Telephone*. El primero y más entusiasta investigador de la teoría de análisis de circuitos de estructura general es Gabriel Kron (ref 1), quien redujo el análisis de circuitos a una serie de operaciones rutinarias que eliminaban el tener que "pensar" y, por lo tanto, se podían automatizar. Desgraciadamente expresó sus trabajos en términos de conceptos matemáticos no muy comunes en la rama de ingeniería, por lo que pocos entienden sus métodos. Fue necesario que otros investigadores, como Le Corbeiller y Branin (refs 2 y 3), trascribieran sus trabajos a términos más comunes en ingeniería.

Desde 1958, veinte años después de los esfuerzos iniciales de Kron, la rama de análisis de circuitos con computadora ha crecido a velocidad vertiginosa. Hoy existen cientos de programas para analizar circuitos lineales y no lineales en el dominio del tiempo y la frecuencia. Muchos de ellos dan información

*Instituto de Ingeniería, UNAM, México, D. F.
**Universidad de California, Los Angeles, E.U.A.

sobre sensibilidad, variaciones estadísticas, y otras cantidades de interés. Se usa una gran variedad de técnicas, entre ellas, métodos de nudos y de mallas, métodos de variables de estado, topológicos y de manipulación de cadenas de símbolos.

El pionero del método de variables de estado en el análisis de circuitos lineales fue Bashkow (ref 4), quien introdujo a la teoría de circuitos conceptos en uso en el campo de ecuaciones diferenciales; Bryant (ref 5), Branin (ref 6), Dervisoglu (ref 7) y otros, ampliaron sus ideas. El método de variables de estado es la base matemática de muchos de los programas existentes para análisis transitorio.

En el análisis de circuitos estáticos no lineales, el trabajo teórico fundamental se debe a Duffin (ref 8), Minty (ref 9) y Katzenelson (ref 10), y las extensiones a circuitos no lineales RLC a Stern (ref 11) y Desoer y Katzenelson (refs 12 y 13).

Algunos de los programas de análisis están basados en métodos topológicos. El trabajo fundamental en esta área es obra de Kirchhoff y Maxwell. Percival (refs 14 y 15) nuevamente mostró interés por este tema. Otros contribuyentes son Mason (ref 16), Coates (ref 17) y Mayeda y Van Valkenburg (ref 18).

Los métodos topológicos permiten obtener ecuaciones del circuito en forma simbólica (en vez de obtener sus valores calculados numéricamente en puntos discretos). Su principal desventaja es el tiempo de computación; además, la cantidad de memoria necesaria aumenta sensiblemente con el tamaño del circuito, por lo que solo se pueden usar para circuitos relativamente pequeños.

Un nuevo enfoque del análisis simbólico con computadora digital ha sido dado por Happ (ref 19), quien usa métodos de reogramas (diagramas o gráficas de flujo; del griego, reos: flujo, y gramos: representación). Algunas extensiones al análisis de circuitos no lineales por medio de reogramas han sido hechas por Kobylarz (ref 20), y Roska (ref 21) ha desarrollado un método completamente independiente, que usa números primos para representar los diferentes símbolos, y multiplicación ordinaria en enteros en lugar de manipular exponentes.

El análisis transitorio de circuitos distribuidos se facilita más con técnicas de transformadas de Laplace que con variables de estado. El algoritmo de la transformada rápida de Fourier, de Cooley y Tukey (ref 22) ha sido la aportación más importante a este tema, pues aumenta notablemente la eficiencia de las computaciones. Los métodos de transformadas de Laplace también son convenientes en el análisis de circuitos que contienen "cajas-negras" en el dominio de la frecuencia y el tiempo (ref 23).

Con respecto a los programas generales de análisis de circuitos que se han escrito, es imposible dar una lista completa, pues existen cientos. Entre los más conocidos están ECAP (ref 24), NASAP (ref 25), NET-1 (ref 26), CIRCUS (ref 27) y SCEPTRE (ref 28). Existen trabajos bibliográficos en los que se comparan sus características (refs 29 a 31).

Una vez que se ha resuelto el problema del análisis de circuitos, es posible considerar el de su diseño. Este se puede hacer automáticamente a base de análisis repetidos, mejorando los valores de los parámetros a cada paso con la ayuda de un programa de optimización hasta que la diferencia entre el comportamiento del circuito y el deseado sea mínima. El trabajo básico en cuanto a la optimización de parámetros ha sido hecho por analistas numéricos. Una obra que da una visión panorámica de la materia es la de Spang (ref 32). El estado de los trabajos de optimización para diseño de circuitos hasta 1967 se encuentra en un artículo de Temes y Calahan (ref 33). Desde entonces ha aparecido un método diferente, el llamado inclusión singular (refs 34 y 68); en él, en lugar de optimización se introducen elementos singulares en el circuito para obligar a las variables a asumir los valores deseados para el diseño. Los elementos ajustables se dejan libres y únicamente se escriben las ecuaciones de Kirchhoff. En estas condiciones, el circuito ajusta el valor de sus corrientes y voltajes para satisfacer las demandas impuestas por los elementos singulares. Los valores de los elementos ajustables se determinan al final del proceso, a partir de los voltajes y corrientes del circuito en equilibrio. El método se ha usado para el diseño de problemas de corriente directa, y se han observado mejoras del orden de mil veces en el tiempo de computación.

En el campo de diseño de circuitos con computadoras es también necesario considerar el problema de comunicación entre el hombre y la computadora. El trabajo fundamental en este sentido es obra de Ross y Rodríguez (ref 35), Evans y Katzenelson (ref 36) y So (ref 37).

Desde 1966 han aparecido varios libros sobre diseño de circuitos con computadora (refs 38 a 42), y varias ediciones especiales de revistas de investigación han sido dedicadas a este tema (refs 43 a 45). Además, se ha tratado en numerosos simposios, conferencias y cursos cortos en varios países (ref 46 a 61). Finalmente, se han dedicado sesiones especiales a este tema en innumerables conferencias de ingeniería.

2. EJEMPLOS ILUSTRATIVOS DEL USO DE UN PROGRAMA

A continuación se presenta un ejemplo en que se analiza un circuito mediante un programa de computadora digital. Con él, el lector podrá darse cuenta de cómo se usan dichos programas. El que aquí se ilustra es NASAP, un programa cooperativo desarrollado por varias universidades y que puede ser usado por un ingeniero aun sin conocimiento de programación (ref 62).

Datos al programa. Para que el programa analice un circuito, el usuario necesita describir su topología usando ciertas reglas que se describen más adelante. También es necesario dar los valores de los elementos y las cantidades que se desean calcular.

Resultados del programa. El programa calcula las cantidades especificadas (funciones de transferencia entre cualquier par de variables), así como la sensibilidad con respecto a cualquier parámetro dado por el usuario. Estos dos cálculos pertenecen al dominio de la frecuencia. En cuanto al dominio del tiempo, el programa calcula la respuesta al impulso correspondiente a la función de transferencia deseada.

Preparación del circuito para análisis. El primer paso es elaborar un diagrama esquemático del circuito que se piensa analizar. Por ejemplo, considérese el circuito de la fig 1. Se trata de una etapa de amplificación con un transistor con emisor común, seguido de una red de acoplamiento sintonizada. En la fig 2 se muestra el mismo circuito en forma adecuada para el análisis lineal en el dominio de la frecuencia. Obsérvese que se trata de un análisis incremental y, por lo tanto, la batería luce como un cortocircuito. Además, se ha añadido la carga de 50 ohms que representa el efecto del resto del circuito. Como interesa la ganancia de la corriente de entrada a la de salida, el circuito estará excitado por una fuente de corriente unitaria.

El paso final antes de alimentar el circuito a la computadora consiste en: 1) sustituir los transistores por circuitos equivalentes incrementales; 2) numerar todos los elementos; 3) identificar todas las fuentes controladas de voltaje y corriente; 4) establecer direcciones arbitrarias de flujo de corriente y polaridades de voltaje; 5) dar valores numéricos a todos los elementos y parámetros; 6) numerar todos los nudos consecutivamente.

En la fig 3 se muestra el diagrama del circuito una vez que las operaciones anteriores han sido hechas y marcadas en el esquema.

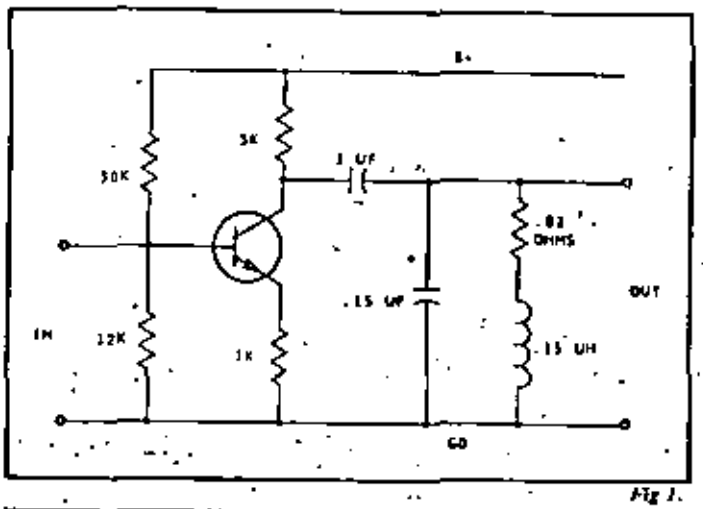


Fig 1.

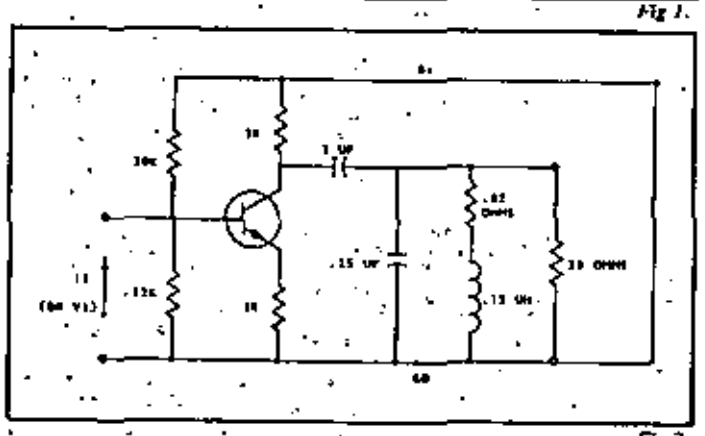


Fig 2.

Alimentación de datos al programa. Una vez que se ha preparado el esquema en la forma indicada, se procede a alimentar la computadora la cual no tiene medios para "ver" el diagrama, por lo que debe describirse en clave. Para ello se siguen ciertas reglas, las cuales se explicarán con base en el ejemplo dado.

La fig 4 muestra el contenido de las tarjetas que alimentan a la computadora para describir el circuito, y los cálculos que se desean hacer. La tarjeta 1 es de control e identificación. Las palabras NASAP PROBLEM indican a la computadora que en las siguientes tarjetas aparecerá la descripción de un circuito que se debe analizar. El resto de la tarjeta tiene títulos cuyo contenido es arbitrario y sirve para facilitar la identificación del circuito al que corresponde un determinado análisis. Las siguientes 15 tarjetas describen el circuito de la fig 3 en forma comprensible para la máquina. El significado de las cantidades allí contenidas es el siguiente:

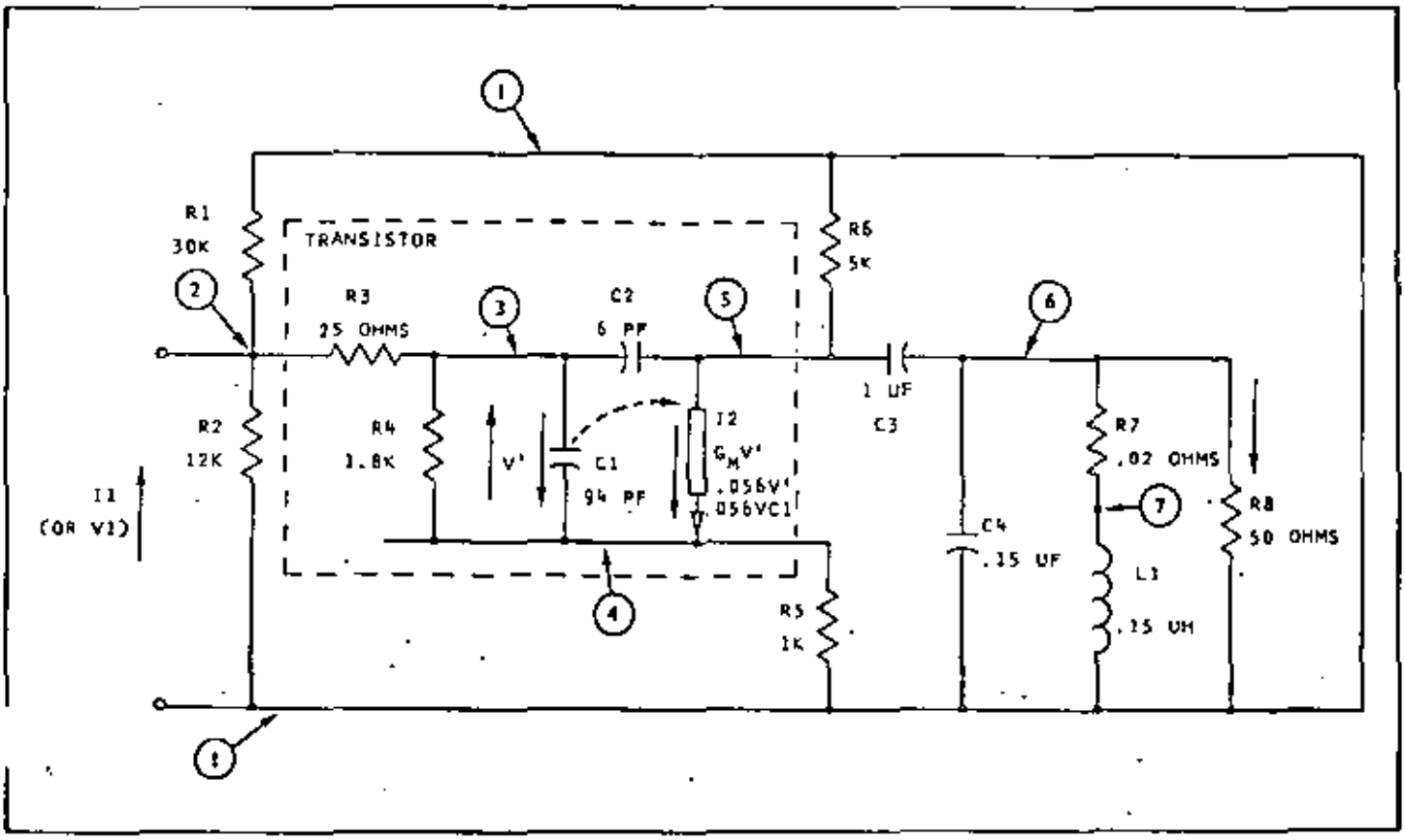


Fig 3.

```
NASAP PROBLEM 41N01    CODING BOOK 011769

I1 1 2 1.
R1 1 2 30K
R2 2 1 12K
R3 2 3 25
R4 3 4 1.0K
C1 3 4 94PF
C2 3 5 6PF
I2 5 4 0.056 VC1
R5 4 1 1K
R6 5 1 5K
C3 5 6 1UF
C4 6 1 0.15UF
R7 6 7 0.02
L1 7 1 0.15UH
R8 6 1 50
OUTPUT
IR8/II1/C4
FREQ  5.8  6.2  0.01
TIME 0.000001
EXECUTION
```

Fig 4.

La primera tarjeta indica la presencia de una fuente de corriente llamada I1, conectada a los nudos 1 y 2 (el orden en que se dan los nudos indica la dirección de referencia), cuyo valor es 1 ampere.

La segunda tarjeta indica la presencia de una resistencia llamada *R1*, conectada a los nudos 1 y 2, cuyo valor es de 30 kilohms.

La tercera, cuarta y quinta dan información similar que el lector puede deducir fácilmente.

La sexta tarjeta indica la presencia de un capacitor *C1* conectado a los nudos 3 y 4, cuyo valor es de 94 picofarads.

La octava tarjeta indica la presencia de una fuente de corriente *I2* que está conectada a los nudos 5 y 4, cuyo valor es 0.056 veces el valor del voltaje a través del capacitor *C1* (indicado por la sigla *VC1*) que la controla.

La decimocuarta tarjeta indica la presencia de un inductor *L1* conectado entre los nudos 7 y 1, cuyo valor es de 0.15 microhenries.

El lector puede comparar el resto de las tarjetas no descritas con el esquema de la fig 3, para darse cuenta cómo se le puede describir un circuito a una máquina "ciega".

Las últimas cinco tarjetas contienen información sobre los cálculos que desea el usuario que haga la computadora. La tarjeta OUTPUT indica a la computadora que ha terminado la descripción del circuito y que a continuación aparecerán los resultados deseados. La siguiente tarjeta indica que se desea calcular la ganancia *IR8/II1* y conocer la sensibilidad de dicha ganancia a variaciones en la capacitancia *C4*. La antepenúltima tarjeta indica que se desea evaluar dichas cantidades en el intervalo de frecuencias entre 10 a la potencia 5.8, y 10 a la potencia 6.2 en 100 (1/.01) intervalos logarítmicos por década. La penúltima indica que se desea la respuesta en el tiempo correspondiente a la función de transferencia indicada en las anteriores tarjetas, y que esta función se debe evaluar hasta .000001 segundos. (El programa da automáticamente 100 puntos intercalados.) La última tarjeta indica que toda la información ha sido dada y ordena a la computadora que ejecute el análisis.

Ejecución del programa. El programa genera en la impresora los siguientes resultados:

1. Una copia de los datos contenidos en las tarjetas que alimentaron a la computadora. (Esto sirve para asegurarse de que la computadora ha aceptado los datos correctamente, y para facilitar la localización de errores en la codificación, en caso de haberlos.)

2. Una lista de los números de mallas de diferentes órdenes en la solución del problema por métodos de reogramas. Esto es de interés para el teórico en reogramas. Los números pueden servir de guía al usuario para darse una idea de la complejidad computacional del problema.

En vista de que lo mencionado en el punto 1 es una copia idéntica de la fig 4, y que los resultados de 2 son de poco interés para el ingeniero diseñador, no se muestran esos resultados.

3. La función de transferencia (o varias de ellas si así se pidió), la cual aparece impresa como una razón de polinomios en la frecuencia compleja.

4. Los ceros y polos de la función de transferencia.

Estos dos últimos aparecen en la fig 5. (Debe hacerse notar que los valores de los polos y los ceros son muy extraños, y están equivocados; se aclara esta situación más adelante. Obsérvense también los mensajes producidos por la computadora sobre desborde numérico)

```
TRANSFER FUNCTION   IR8/II1/C4

[illegible]

OVERFLOW AT 40476 IN AC

OVERFLOW AT 40534 IN AC AND MQ.

OVERFLOW AT 40627 IN AC

OVERFLOW AT 40635 IN AC

OVERFLOW AT 40653 IN AC


ZEROS OF TRANSFER FUNCTION

ZEROS    REAL PART     IMAG. PART

  1   0.15000E 00   0.00000E-38
  2   0.15000E 00   0.00000E-38
  3   0.15000E 00   0.00000E-38
  4   0.15000E 00   0.00000E-38


POLES OF TRANSFER FUNCTION

POLES    REAL PART     IMAG. PART

  1   0.15000E 00   0.00000E-38
  2   0.15000E 00   0.00000E-38
  3   0.15000E 00   0.00000E-38
  4   0.15000E 00   0.00000E-38
  5   0.15000E 00   0.00000E-38
```

Fig 5.

5. Varias listas en las que aparecen el logaritmo de la frecuencia, la frecuencia, la parte real e imaginaria de la función de transferencia *H(S)*, los logaritmos de las anteriores, la magnitud y fase, el logaritmo de la magnitud, y la magnitud en decibeles de la función *H(S)*. En la fig 6 aparece una lista parcial que contiene algunas de las cantidades mencionadas. Además de obtener estas cantidades en forma de listas, el programa genera los resultados en gráficas hechas con la impresora. Dos de dichas gráficas en las que aparecen la magnitud de la función de transferencia en decibeles y su fase se muestran en las figs 7 y 8.

6. Se generan varias clases de sensibilidad: logarítmica, que se representa con SENS (H), incremental de 1 por ciento de la parte real, parte imaginaria, magnitud y fase de la función de transferencia, que se representan con SENS (RE(H)), SENS (IM(H)), SENS (ABS(H)), y SENS (PHI(H)), respectivamente. Se generan también sensibilidades de las posiciones de los polos y ceros de la función de transferencia. Estas sensibilidades aparecen en una multitud de formas análogas a las mencionadas en el inciso 5. Asimismo se generan con la impresora las gráficas correspondientes. En las figs 9 a 12 aparecen una tabla parcial y tres gráficas con información referente a las sensibilidades del ejemplo que se está ilustrando.

.. Respuesta transitoria al impulso correspondiente a la función de transferencia, la cual se calcula desde el tiempo cero hasta el instante elegido por el usuario.

Para obtener la respuesta al impulso, el programa hace uso de los polos de la función de transferencia. Como se indicó anteriormente, debido a los desbordes numéricos ocurridos, dichos polos (y ceros) están incorrectamente calculados. La razón por la cual se presenta este problema numérico es la enorme diferencia entre las magnitudes de los coeficientes de los polinomios del numerador y denominador de la función de transferencia que aparece en la fig 5. La dificultad se puede vencer en parte si se hace un cambio de variables $s' = 100\,000s$. El efecto de este cambio en los valores de los elementos del circuito es el de multiplicar por 100 000 los valores de las inductancias y capacitancias, dejando todos los demás parámetros del circuito sin cambio. Con respecto al tiempo, el efecto es multiplicarlo por 100 000. Finalmente, habrá también que multiplicar el valor del fenómeno transitorio por el mismo factor de 100 000; todas estas equivalencias son bien conocidas en teoría de sistemas lineales (ref 63).

En la fig 13 se muestra la descripción del circuito tras el cambio de variables. Obsérvese que las resistencias y transconductancias en el modelo del transistor permanecen iguales mientras que las inductancias y capacitancias han sido multiplicadas por 100 000. Se hace notar, igualmente, el cambio en los datos

LOG(FREQ)	FREQ	RE(H)	IM(H)	LOG(RE(H))	LOG(IM(H))
0.500000E 01	0.453959E 06	-0.454956E-01	-0.129491E 00	-0.134203E 01	-0.888684E 00
0.501000E 01	0.465654E 06	-0.466873E-01	-0.132811E 00	-0.133254E 01	-0.876425E 00
0.502000E 01	0.476604E 06	-0.522248E-01	-0.142557E 00	-0.128223E 01	-0.846009E 00
0.503000E 01	0.476628E 06	-0.561643E-01	-0.149706E 00	-0.125055E 01	-0.825175E 00
0.504000E 01	0.691438E 06	-0.688748E-01	-0.157557E 00	-0.121771E 01	-0.802564E 00
0.505000E 01	0.707945E 06	-0.655452E-01	-0.166217E 00	-0.118349E 01	-0.779532E 00
0.506000E 01	0.724435E 06	-0.711696E-01	-0.175428E 00	-0.114770E 01	-0.756011E 00
0.507000E 01	0.741103E 06	-0.775986E-01	-0.185654E 00	-0.111014E 01	-0.731916E 00
0.508000E 01	0.758577E 06	-0.850041E-01	-0.196476E 00	-0.107054E 01	-0.707191E 00
0.509000E 01	0.776246E 06	-0.936108E-01	-0.212357E 00	-0.102863E 01	-0.672953E 00
0.510000E 01	0.793280E 06	-0.103755E 00	-0.224157E 00	-0.984398E 00	-0.649564E 00
0.511000E 01	0.812803E 06	-0.115042E 00	-0.246231E 00	-0.939131E 00	-0.608857E 00
0.512000E 01	0.831763E 06	-0.131486E 00	-0.267528E 00	-0.880453E 00	-0.572629E 00
0.513000E 01	0.851137E 06	-0.148587E 00	-0.292497E 00	-0.828017E 00	-0.533359E 00
0.514000E 01	0.870963E 06	-0.171415E 00	-0.323454E 00	-0.765944E 00	-0.490186E 00
0.515000E 01	0.891250E 06	-0.201187E 00	-0.361281E 00	-0.696525E 00	-0.442250E 00
0.516000E 01	0.912810E 06	-0.241203E 00	-0.408479E 00	-0.617609E 00	-0.389050E 00
0.517000E 01	0.933254E 06	-0.297989E 00	-0.470444E 00	-0.525914E 00	-0.327122E 00
0.518000E 01	0.954992E 06	-0.383471E 00	-0.554123E 00	-0.416264E 00	-0.256393E 00
0.519000E 01	0.977237E 06	-0.524870E 00	-0.669999E 00	-0.279947E 00	-0.173928E 00
0.520000E 01	0.999999E 06	-0.791972E 00	-0.832668E 00	-0.101284E 00	-0.795777E-01
0.521000E 01	0.102329E 07	-0.149527E 01	-0.100858E 01	0.174834E 00	0.371318E-02
0.522000E 01	0.104712E 07	-0.294199E 01	-0.569267E 00	0.468641E 00	-0.244975E 00
0.523000E 01	0.107151E 07	-0.195356E 01	0.251642E 01	0.290824E 00	0.400285E 00
0.524000E 01	0.109678E 07	-0.154184E 01	0.183945E 01	0.188154E 00	0.264584E 00
0.525000E 01	0.112201E 07	0.154644E 00	0.118332E 01	-0.810661E 00	0.730368E-01
0.526000E 01	0.114815E 07	0.202171E 00	0.842284E 00	-0.694280E 00	-0.743165E-01
0.527000E 01	0.117489E 07	0.199713E 00	0.646025E 00	-0.699594E 00	-0.189753E 00
0.528000E 01	0.120224E 07	0.186708E 00	0.519920E 00	-0.729844E 00	-0.284633E 00
0.529000E 01	0.123028E 07	0.172388E 00	0.452743E 00	-0.763499E 00	-0.344738E 00
0.530000E 01	0.125895E 07	0.159107E 00	0.368965E 00	-0.798316E 00	-0.433014E 00
0.531000E 01	0.128824E 07	0.147345E 00	0.320374E 00	-0.831663E 00	-0.494342IE 00
0.532000E 01	0.131825E 07	0.137068E 00	0.282113E 00	-0.863061E 00	-0.549578E 00
0.533000E 01	0.134895E 07	0.128095E 00	0.251196E 00	-0.892965E 00	-0.599868E 00
0.534000E 01	0.138038E 07	0.120229E 00	0.225682E 00	-0.919989E 00	-0.646503E 00
0.535000E 01	0.141253E 07	0.113291E 00	0.204259E 00	-0.945801E 00	-0.689876E 00
0.536000E 01	0.144543E 07	0.107132E 00	0.185493E 00	-0.970077E 00	-0.731853E 00
0.537000E 01	0.147910E 07	0.101628E 00	0.170235IE 00	-0.992988E 00	-0.768910E 00
0.538000E 01	0.151356IE 07	0.966759E-01	0.156467E 00	-0.101548E 01	-0.805519E 00
0.539000E 01	0.154881E 07	0.921925E-01	0.144302E 00	-0.103534E 01	-0.840962E 00

Fig 6.

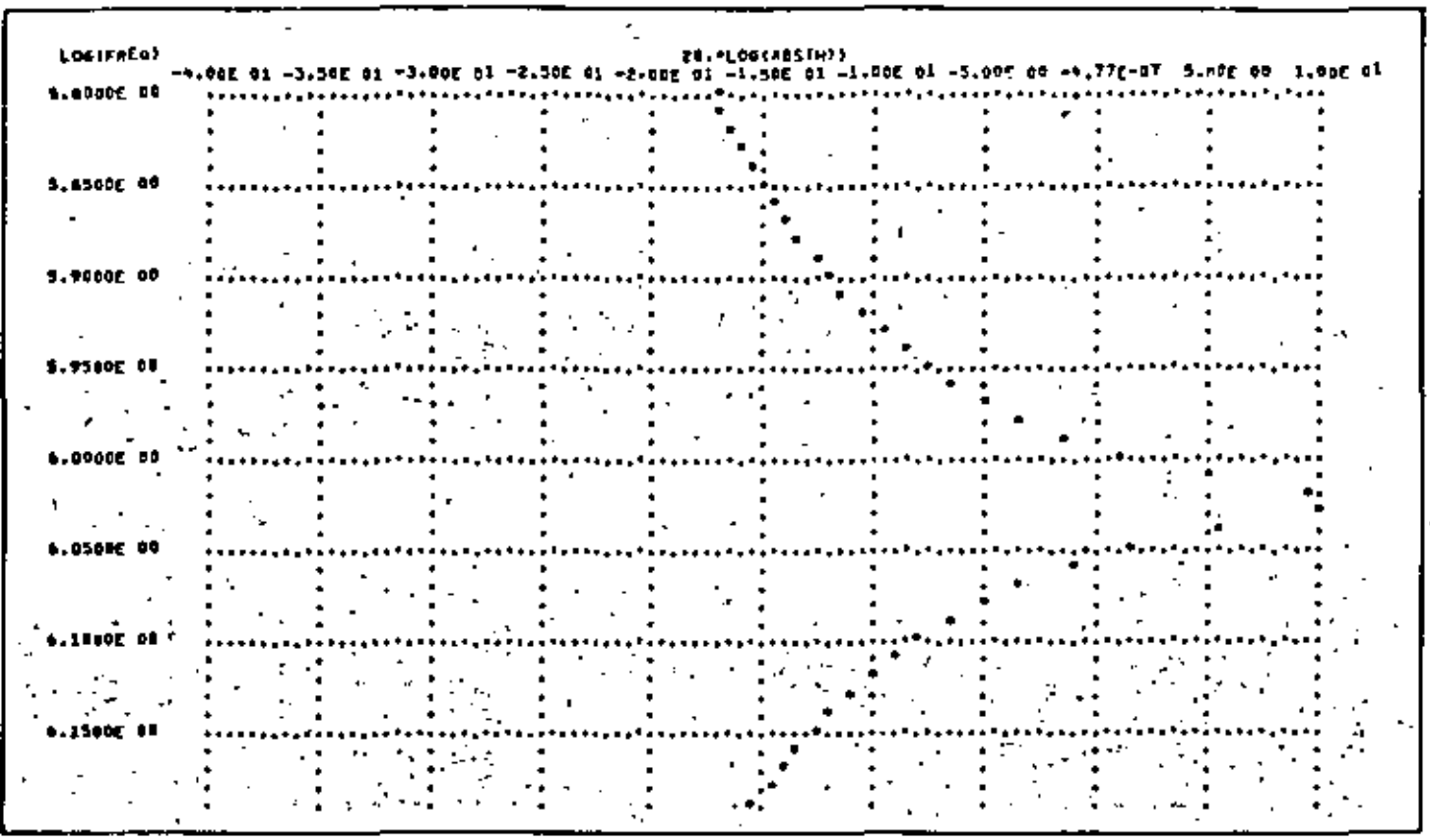

LOG(FREQ)
20.*LOG(ABS(H))
-4.00E 01 -3.50E 01 -3.00E 01 -2.50E 01 -2.00E 01 -1.50E 01 -1.00E 01 -5.00E 00 -4.77E-07 5.00E 00 1.00E 01
5.8000E 00
5.8500E 00
5.9000E 00
5.9500E 00
6.0000E 00
6.0500E 00
6.1000E 00
6.1500E 00

Fig 7.

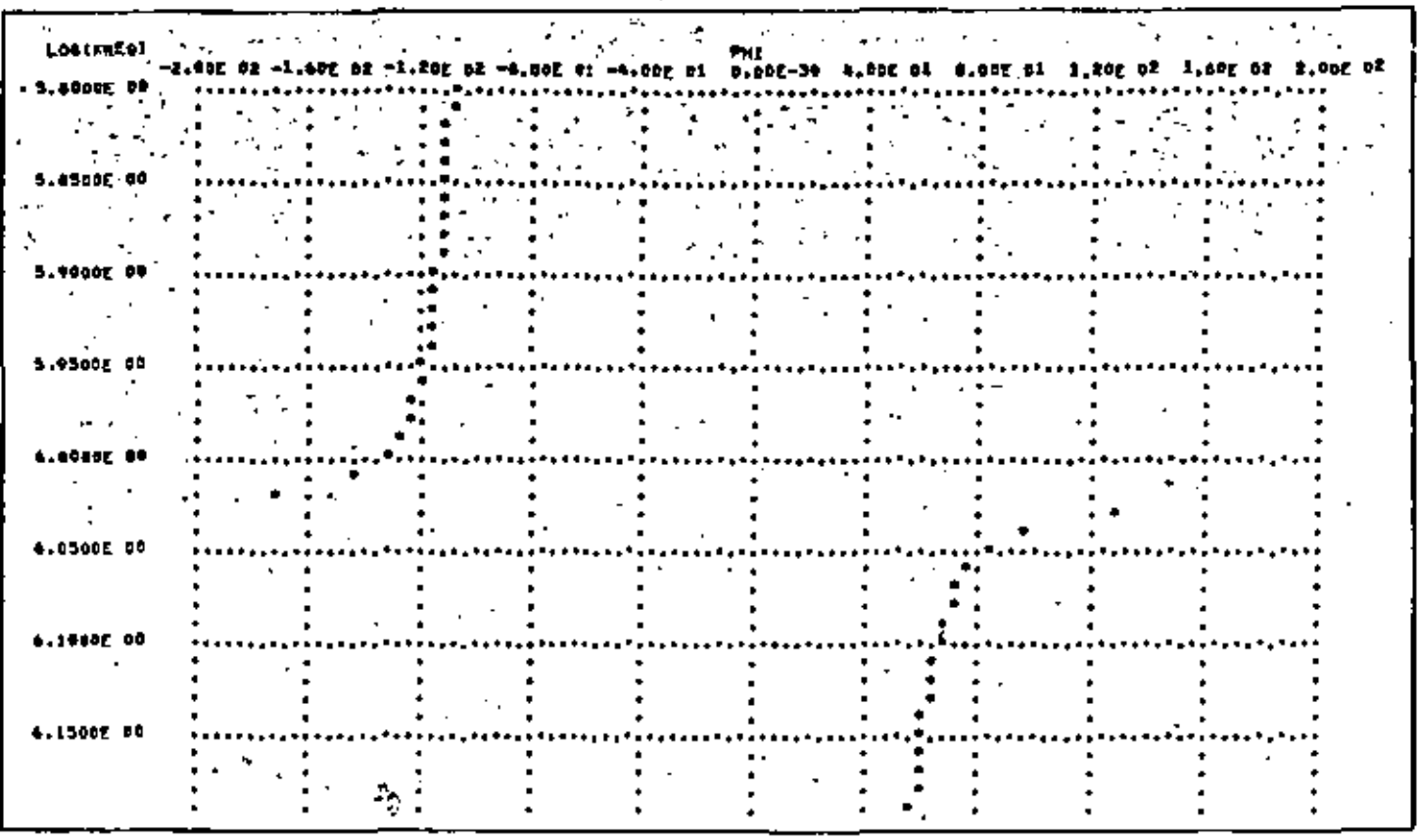
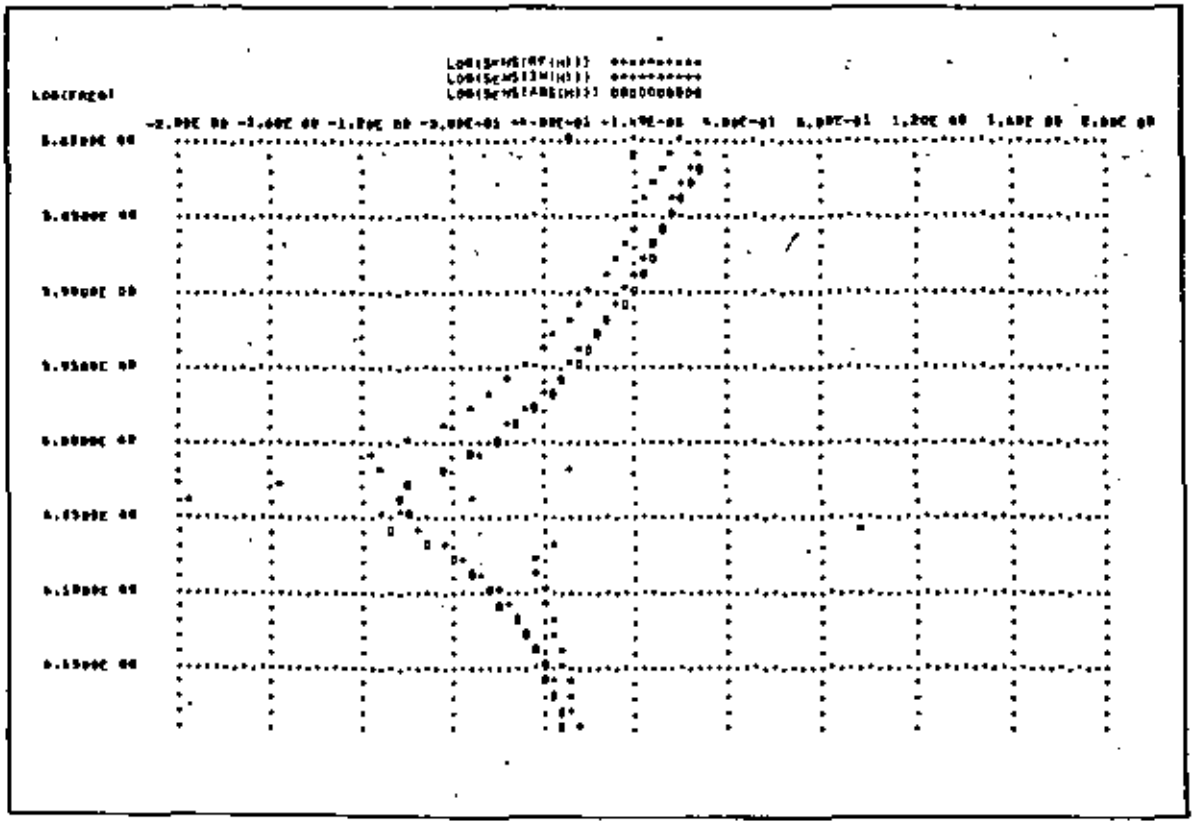

LOG(FREQ)
PHI
-2.40E 02 -2.00E 02 -1.60E 02 -1.20E 02 -8.00E 01 -4.00E 01 0.00E-30 4.00E 01 8.00E 01 1.20E 02 1.60E 02 2.00E 02
5.8000E 00
5.8500E 00
5.9000E 00
5.9500E 00
6.0000E 00
6.0500E 00
6.1000E 00
6.1500E 00

Fig 8.

LOG(FREQ)	FREQ	LOG(SENS(Rg(N)))	LOG(SENS(Wg(N)))	LOG(SENS(BS(N)))	LOG(SENS(PNT(N)))
0.5A0000DE 01	0.6389572E 06	0.1853259E 00	0.2791566E 00	0.2632713E 00	0.1694977E 01
0.5C1000DE 01	0.6456541E 06	0.1525709E 00	0.2433190E 00	0.2320041E 00	0.1657690E 01
0.5E2000DE 01	0.6606933E 06	0.1283320E 00	0.2116821E 00	0.1998867E 00	0.1619473E 01
0.5F3000DE 01	0.6760628E 06	0.8880793E-01	0.1791664E 00	0.1668288E 00	0.1579243E 01
0.5*40000E 01	0.6918308E 06	0.5398100E-01	0.1456775E 00	0.1327281E 00	0.1536790E 01
0.5*50000E 01	0.7079*40E 06	0.1861431E-01	0.1111074E 00	0.9745407E-01	0.1491872E 01
0.5*60000E 01	0.724*357E 06	-0.1823515E-01	0.7533100E-01	0.6087309E-01	0.1444208E 01
0.5A7000DE 01	0.7413104E 06	-0.5841374E-01	0.3M2007AE-01	0.2261234E-01	0.1393451E 01
0.5A8000DE 01	0.7585775E 06	-0.9729647E-01	-0.4548371E-03	-0.1692912E-01	0.1339212E 01
0.5A9000DE 01	0.7762*69E 06	-0.1369601E 00	-0.4084108E-01	-0.5859545E-01	0.1281086E 01
0.5**8000E 01	0.794332A0E 06	-0.1852C59E 00	+0.8320206E-01	-0.1029*89E 00	0.1218244E 01
0.5*10000E 01	0.8128365E 06	-0.2333985E 00	-0.1278321E 00	-0.1498962E 00	0.1150196E 01
0.5*20000E 01	0.8317635E 06	-0.2858934E 00	+0.1750670E 00	-0.1985038E 00	0.1075943E 01
0.5930920E 01	0.8511379E 06	-0.3*09794E 00	+0.2254087E 00	-0.2516936E 00	0.9953066E 00
0.5**40000E 01	0.870*634E 06	-0.*019585E 00	-0.2792977E 00	+0.3093142E 00	0.9017391E 00
0.5*50000E 01	0.8912507E 06	-0.*60239*E 00	-0.3374023E 00	-0.372*095E 00	0.8021547E 00
0.5960000E 01	0.9120106E 06	-0.5***A50E 00	-0.*0C*082E 00	+0.**24176E 00	0.6866504E 00
0.5*70000E 01	0.9332541E 06	-0.6300**3E 00	-0.*689082E 00	-0.5213807E 00	0.5530317E 00
0.5*80000E 01	0.9549923E 06	-0.7295139E 00	-0.5426711E 00	-0.6122870E 00	0.3999483E 00
0.5*90000E 01	0.9772370E 06	-0.8*72853E 00	-0.6373933E 00	+0.7195742E 00	0.2422*28E 00
0.6000000E 01	0.9999***8E 06	-0.9*68377E 00	-0.7650342E 00	-0.8*92724E 00	-0.*819219E-01
0.*01*000E 01	0.1023293E 07	-0.1155*0*E 01	+0.2747742E 00	-0.1001622E 01	-0.3**0283E 00
0.*02000DE 01	0.1047128E 07	-0.1319*25E 01	-0.2079264E 01	-0.1036*43E 01	-0.7*801*1E 00
0.*03000DE 01	0.1071519E 07	-0.1557*8*E 01	-0.7290945E 00	-0.1010*67E 01	-0.*5*0612E 00
0.*0*000DE 01	0.1096*78E 07	-0.1*67134E 01	-0.1*41150E 01	-0.1063097E 01	-0.*138758E 00
0.*C5000DE 01	0.1122815E 07	-0.112*68*E 01	-0.*5**42*1E 00	-0.*257545E 00	-0.2*82*07E 00
0.*60000DE 01	0.1148157E 07	0.*629358E 00	-0.*3*5*8*E 00	-0.*103945E 00	-0.5*58*9E-01
0.*07000DE 01	0.1174*67E 07	-0.3*371349E 00	+0.7*25377E 00	-0.7191575E 00	0.1323368E 00
0.*08000DE 01	0.1202284E 07	-0.*2*8136E 00	-0.*672*07E 00	+0.**549*9E 00	0.2*76*21E 00
0.*09000DE 01	0.1230268E 07	-0.*51*363E 00	-0.5**528E 00	-0.5*852**E 00	0.*184695E 00
0.*10000DE 01	0.1258925E 07	-0.*179067E 00	+0.551*126E 00	-0.5330068E 00	0.5311144E 00
0.*11000DE 01	0.1288249E 07	-0.3979520E 00	-0.5057316E 00	+0.4887393E 00	0.*2*7456E 00
0.*12000DE 01	0.1318257E 07	-0.3762562E 00	-0.*A5**62E 00	-0.*501844E 00	0.7172505E 00
0.*13000DE 01	0.1348963E 07	-0.3545*0*E 00	+0.*30*270E 00	+0.*162347E 00	0.7*57029E 00
0.*1*000DE 01	0.1380384E 07	-0.3327*13E 00	-0.3*9*261E 00	-0.2886675E 00	0.*466*29E 00
0.*150000E 01	0.1412537E 07	-0.3*40956E 00	-0.3720027E 00	-0.3590572E 00	0.9312*32E 00
0.*160000E 01	0.1445439E 07	-0.295627*E 00	-0.3*69*67E 00	+0.33*7176E 00	0.*90*164E 00
0.*170000E 01	0.1479108E 07	-0.27*3648E 00	-0.32*7602E 00	-0.3126639E 00	0.1***887E 01
0.*180000E 01	0.1513561E 07	-0.2622892E 00	-0.3036207E 00	-0.29258*0E 00	0.10*52*8E 01
0.*190000E 01	0.1548816E 07	-0.2*725*1E 00	-0.2**7655E 00	-0.27*2313E 00	0.11*1*52E 01

Fig 9.

Fig 10.



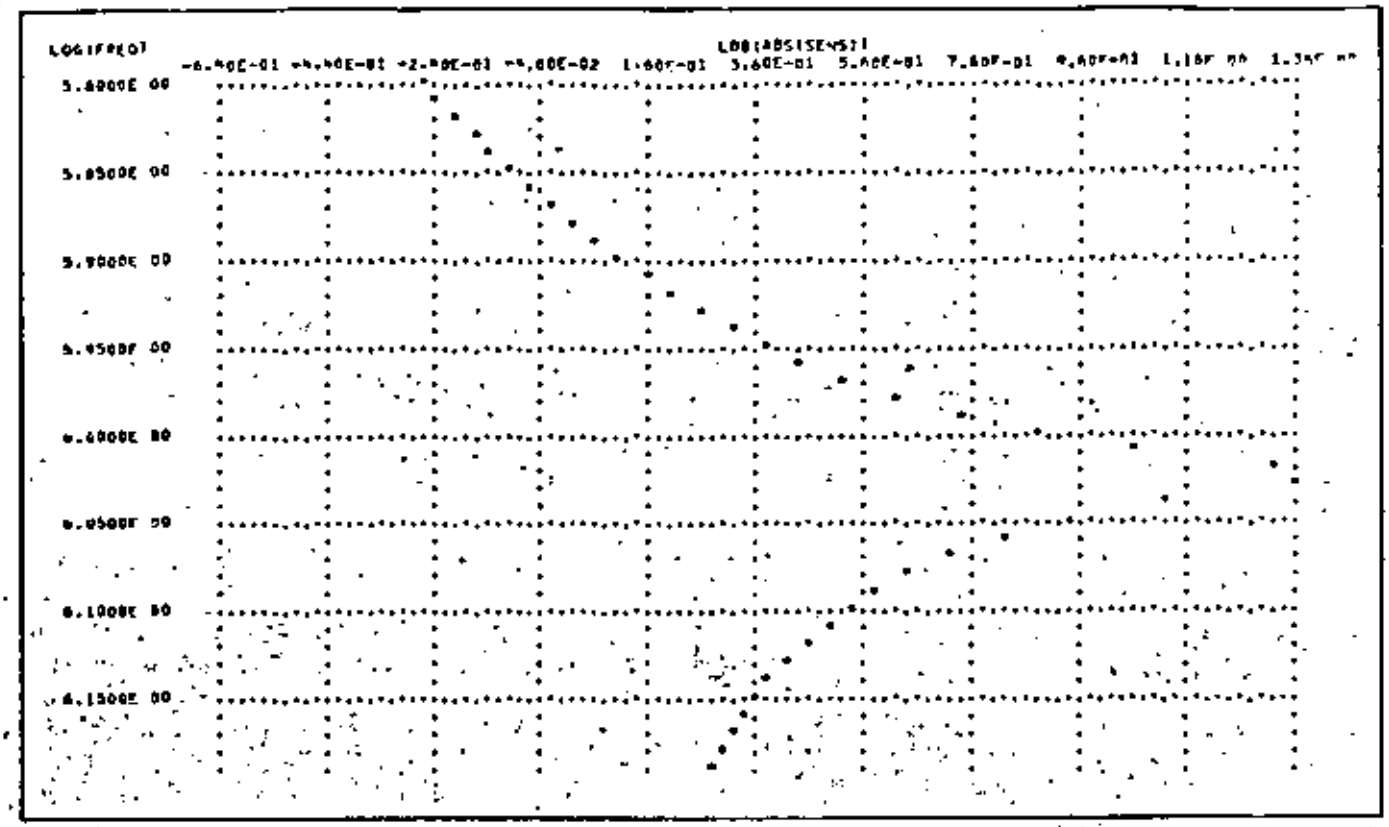
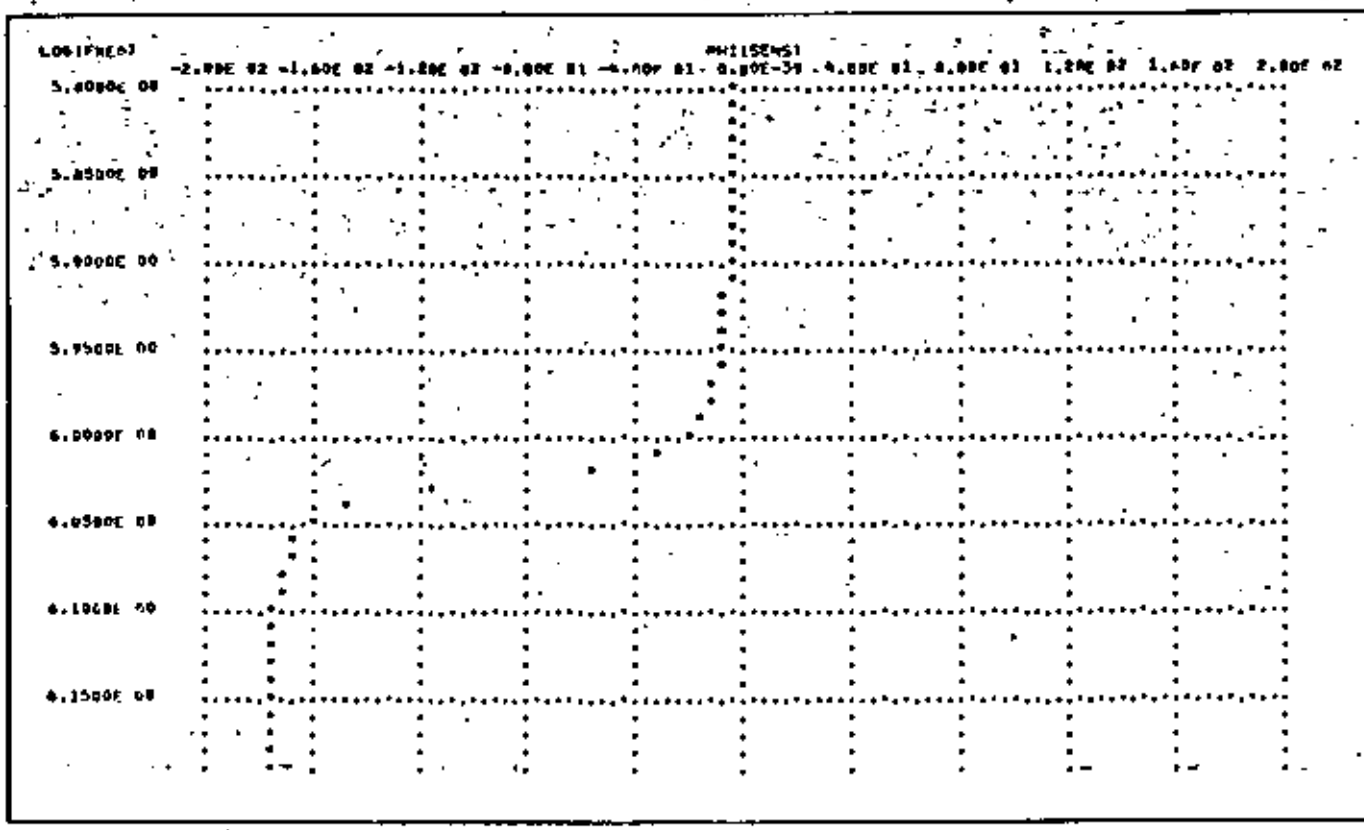

LOG(FREQ)
LOG(ABSISE+S)
5.8000E 00
5.8500E 00
5.9000E 00
5.9500E 00
6.0000E 00
6.0500E 00
6.1000E 00
6.1500E 00

Fig 11.

Fig 12.


LOG(FREQ)
PHI(SE+S)
5.8000E 00
5.8500E 00
5.9000E 00
5.9500E 00
6.0000E 00
6.0500E 00
6.1000E 00
6.1500E 00

```
NASAP PROBLEM 41N02

I1 1 2 1.
R1 1 2 30K
R2 2 1 12K
R3 2 3 25
R4 3 4 1.8K
C1 3 4 9.4UF
C2 3 5 0.6UF
I2 5 4 0.056 VC1
R5 4 1 1K
R6 5 1 5K
C3 5 6 0.1F
C4 6 1 0.015F
R7 6 7 0.02
L1 7 1 0.015H
R8 6 1 50
OUTPUT
IR8/I11/C4
FREQ  0.8  1.2  0.01
TIME 4
EXECUTION

ZEROS OF TRANSFER FUNCTION

ZEROS   REAL PART    IMAG. PART

   1 -0.27605E-10 -0.00000E-38
   2 -0.13333E 01 -0.00000E-38
   3  0.13319E 04  0.00000E-38
   4 -0.74548E 04 -0.00000E-38


POLES OF TRANSFER FUNCTION

POLES   REAL PART    IMAG. PART

   1 -0.19999E-02 -0.00000E-38
   2 -0.13339E 01  0.66656E 02
   3 -0.13339E 01 -0.66656E 02
   4 -0.16448E 03 -0.74186E-11
   5 -0.78151E 04  0.15574E-09
```

Fig 13.

sobre las frecuencias deseadas. Como deben dividirse entre 100 000 y están expresadas en logaritmos base 10, se les ha restado 5 (pues 10^5 = 100 000). Se ve que también aumentó el tiempo de solución a 4 seg (aun cuando el estrictamente equivalente sería 1 seg).

La nueva función de transferencia se muestra en la fig 14. Obsérvese que los coeficientes de los polinomios difieren en magnitud menos que los originales. En la nueva solución no aparecen mensajes de desborde numérico, por lo tanto, puede suponerse que los valores de los polos y ceros son correctos.

En la fig 15 se muestra la sensibilidad de las posiciones de dichos polos y ceros a cambios en *C4*. En la fig 16 se muestra en forma simbólica la respuesta transitoria cuando la excitación es un impulso. Las funciones complejas se pueden combinar con sus conjugadas para producir senoides amortiguadas reales. En la fig 17 la respuesta ha sido evaluada e impresa para 50 puntos intermedios entre cero y 4 seg. En la fig 18 se muestra una gráfica de la respuesta transitoria producida por el programa. (Las líneas rectas han sido trazadas a mano.)

3. DISCUSION

El programa aquí presentado puede ser muy útil en el diseño de circuitos. Para dicho diseño se comienza con un circuito inicial con valores aproximados para los parámetros. Subsecuentemente, con la intuición y experiencia del diseñador, se cambian dichos parámetros analizando los resultados en cada cambio con un programa como NASAP, hasta que el circuito satisfaga los requerimientos del diseño.

Aunque solamente una función de transferencia ha sido exhibida en el ejemplo dado, se pueden obtener tantas como se desee. Por lo tanto, a base de aplicar el programa repetidas veces, se pueden diseñar circuitos multiterminales con caracte-

```
TRANSFER FUNCTION   IR8/I11/C4

                ( 
                ( -3.13E-04     -1.32E 07 S   -9.92E 06 S^2  +6.12E 03 S^3  +1.00E 00 S^4 )
HIS)= 1.29E 00*-----------------------------------------------------------------------------
                (
                (  1.14E 07     +5.72E 09 S   +3.89E 07 S^2  +1.31E 06 S^3  +7.99E 03 S^4  +1.00E 00 S^5 )
```

Fig 14.

SENSITIVITIES OF ZEROES AND POLES OF TRANSFER FUNCTION

ZEROS	REAL	IMAG	SENSITIVITY REAL	SENSITIVITY IMAG
1	-0.2360492E-10	-0.0000000E-38	-0.1768913E-10	-0.0000000E-38
2	-0.1333333E 01	-0.0000000E-38	0.9789552E-08	-0.0000000E-38
3	0.1331899E 04	0.0000000E-38	-0.4036591E-08	-0.0000000E-38
4	-0.7454830E 04	-0.0000000E-38	-0.1180041E-07	-0.0000000E-38

POLES	REAL	IMAG	SENSITIVITY REAL	SENSITIVITY IMAG
1	-0.1999889E-02	-0.0000000E-38	0.1862236E-06	0.0000000E-38
2	-0.1343878E 01	0.6665649E 02	-0.4998695E 00	-0.1320475E-03
3	-0.1343879E 01	-0.6665649E 02	-0.4998694E 00	0.1320352E-03
4	-0.1644778E 03	-0.7418621E-11	-0.2587003E-03	0.6172105E-14
5	-0.7819076E 04	0.1557439E-09	-0.2227968E-05	0.1908007E-17

Fig 15.

IMPULSE RESPONSE FUNCTION

F(T) =

(0.6148E-05 J 0.3694E-13) E (-0.2000E-02 J-0.0000E-38) T

(-0.4349E 01 J 0.2033E 01) E (-0.1344E 01 J 0.6666E 02) T

(-0.4349E 01 J-0.2033E 01) E (-0.1344E 01 J-0.6666E 02) T

(0.9954E 01 J-0.3045E-07) E (-0.1645E 03 J-0.7419E-11) T

(0.7405E-01 J 0.2476E-10) E (-0.7819E 04 J 0.1557E-09) T

Fig 16.

IMPULSE RESPONSE

TIME	IR8/I11
0.0000E-38	0.13294560E 01
0.8000E-01	-0.15681442E 01
0.1600E 00	0.53783759E 01
0.2400E 00	0.68787581E 01
0.3200E 00	0.28423108E 01
0.4000E 00	-0.25810588E 01
0.4800E 00	-0.49865184E 01
0.5600E 00	-0.31232734E 01
0.6400E 00	0.76165859E 00
0.7200E 00	0.33140278E 01
0.8000E 00	0.28449155E 01
0.8800E 00	0.29670326E 00
0.9600E 00	-0.19847885E 01
0.1040E 01	-0.23110320E 01
0.1120E 01	-0.81148662E 00
0.1200E 01	0.10168736E 01
0.1280E 01	0.17159090E 01
0.1360E 01	0.97095013E 00
0.1440E 01	-0.37043358E 00
0.1520E 01	-0.11697519E 01
0.1600E 01	-0.92222770E-00
0.1680E 01	-0.19190023E-01
0.1760E 01	0.72377403E 00
0.1840E 01	0.77096238E 00
0.1920E 01	0.22099977E 00
0.2000E 01	-0.39111479E 00
0.2080E 01	-0.58648539E 00
0.2160E 01	-0.29672897E 00
0.2240E 01	0.16329243E 00
0.2320E 01	0.40976927E 00
0.2400E 01	0.29602532E 00
0.2480E 01	-0.21491611E-01
0.2560E 01	-0.26117987E 00
0.2640E 01	-0.25528330E 00
0.2720E 01	-0.55813360E-01
0.2800E 01	0.14763855E 00
0.2880E 01	0.19912565E 00
0.2960E 01	0.88779057E-01
0.3040E 01	-0.67926929E-01
0.3120E 01	-0.14250045E 00
0.3200E 01	-0.93953517E-01
0.3280E 01	0.16865605E-01
0.3360E 01	0.93384872E-01
0.3440E 01	0.83877894E-01
0.3520E 01	0.12239669E-01
0.3600E 01	-0.54869119E-01
0.3680E 01	-0.67139647E-01
0.3760E 01	-0.25822844E-01
0.3840E 01	0.27200426E-01
0.3920E 01	0.49223371E-01
0.4000E 01	0.29446517E-01

Fig 17.

rísticas en los puertos especificadas por el diseñador. Si estas características se escogen adecuadamente, se pueden obtener, para un circuito cualquiera, las matrices de admitancia o impedancia indefinidas. Una vez que se tienen estas matrices, es posible iniciar el diseño de circuitos microelectrónicos formados por varios subcircuitos multiterminales interconectados. Un programa que hace todo esto automáticamente es el BELNAP (ref 23); con él se pueden analizar circuitos microelectrónicos muy extensos, a condición de hacerlo por partes (ref 64).

El mismo programa es útil para el diseño de circuitos de microondas (ref 67). También se puede usar dicho programa, haciendo combinaciones, para determinar fallas internas en un circuito multiterminal. Para este propósito se han desarrollado métodos matemáticos adecuados (ref 65). Cuando se trata de circuitos con relativamente pocas terminales, es más adecuado el método del cálculo de las desviaciones en lugar de las matrices o los métodos topológicos (ref 66).

No obstante que NASAP y BELNAP son muy poderosos y útiles, no es posible hacer con ellos ciertos tipos de análisis (por ejemplo análisis estadístico o transitorio de circuitos no lineales). Para ello es necesario usar otros programas (refs 24 y 26 a 28), lo cual ocasiona una labor adicional, pues habrá necesidad de codificar el circuito de acuerdo con las reglas especiales de cada programa.

Aunque sería muy conveniente desarrollar un programa para todos los tipos de análisis posibles, esto es sumamente difícil, pues los métodos matemáticos que se aplican en un caso no son válidos para otro. Es mejor desarrollar un compilador especial que sirva de puente entre el usuario y cada programa diferente de análisis. Así, un circuito no tendrá que ser codificado varias veces, aun cuando se requieran diversos tipos de análisis, con lo que se elimina la fuente de muchos errores. El compilador tendrá un idioma universal y traducirá a los idiomas de los programas de análisis. La presentación de resultados se hará en el lenguaje universal tras una traducción de los lenguajes particulares. Dicho compilador lo está desarrollando, en la Universidad de California, en Los Angeles, el coautor de este artículo.

4. FUTURO

El futuro de la rama de diseño de circuitos con computadora y áreas relacionadas, tales como localización de fallas, documentación y producción automática, es sumamente brillante. Con respecto al diseño se nota una fuerte tendencia a utilizar más y

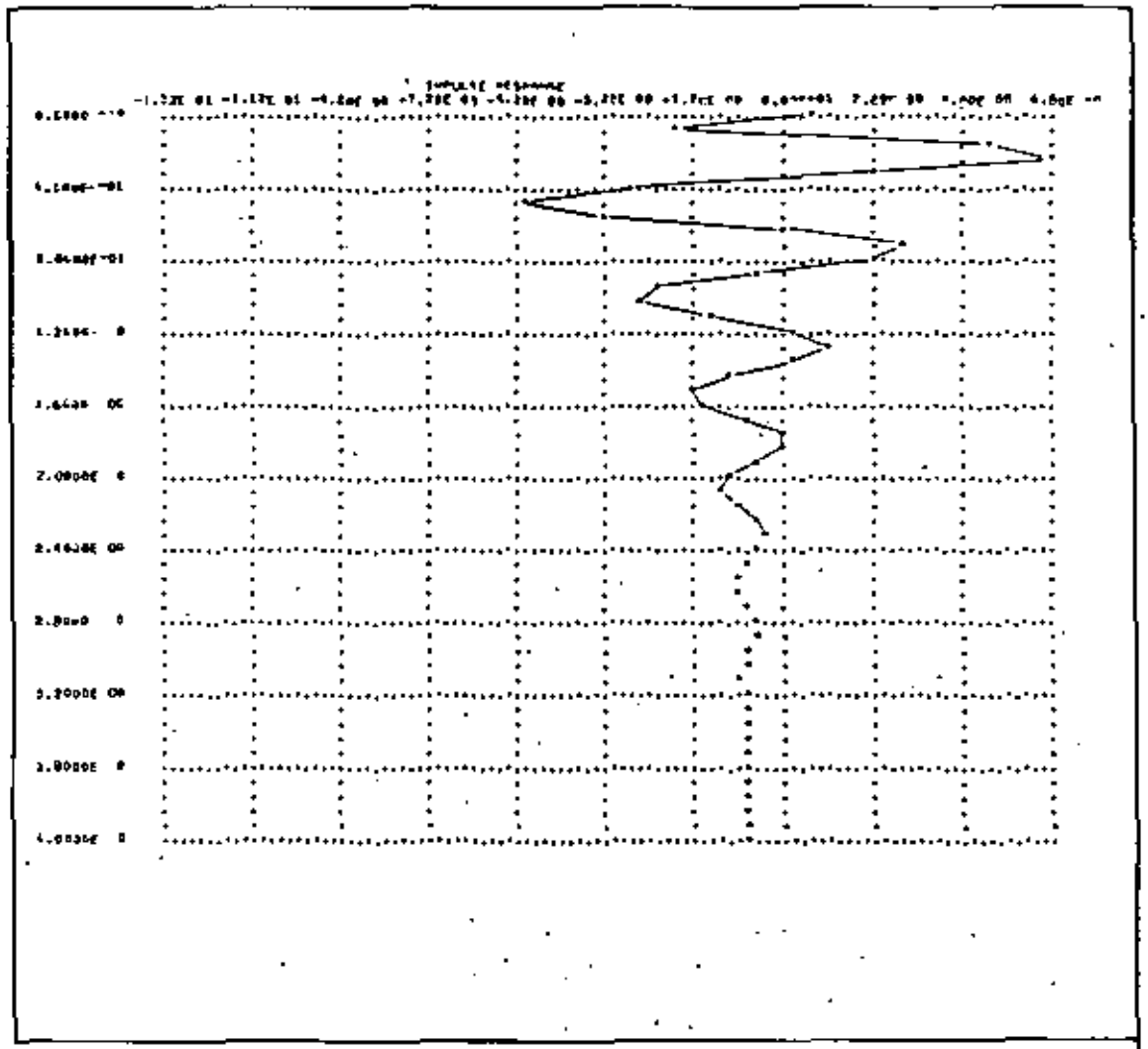


Fig 18.

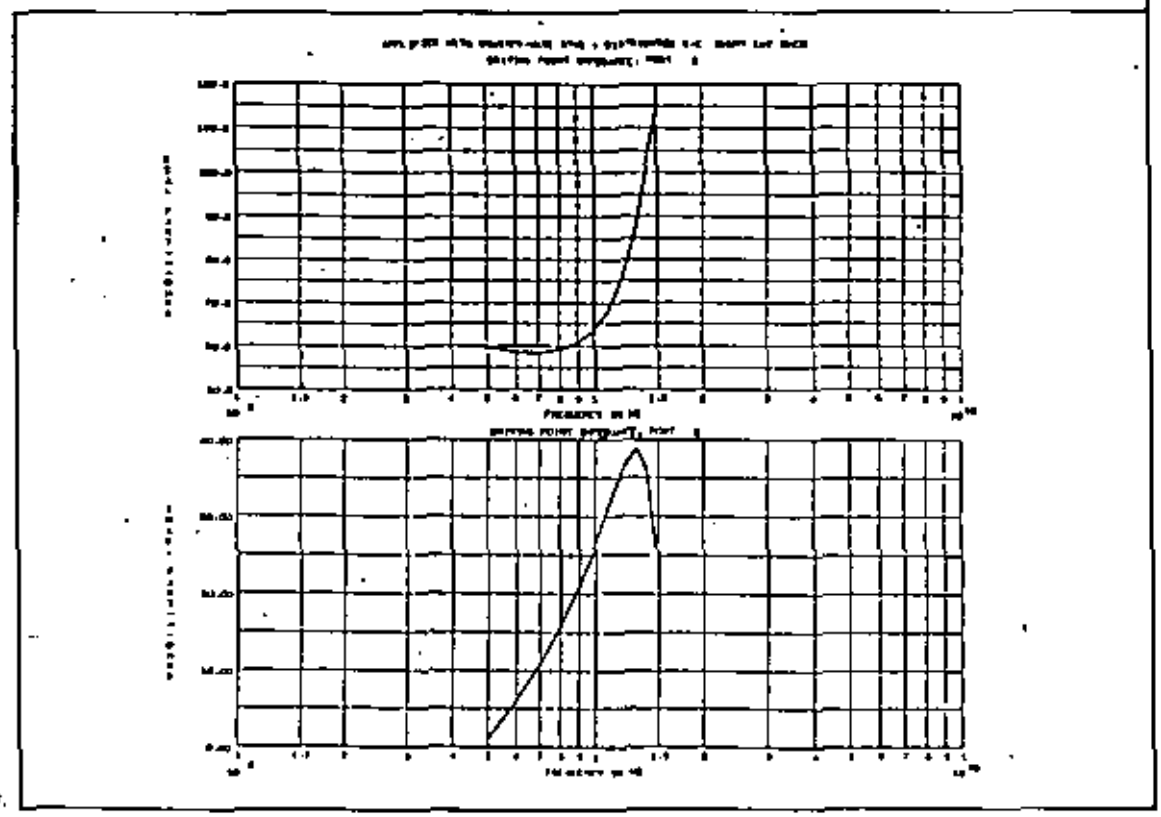


Fig 19.

más el tiempo compartido de operación en lugar de la operación normal. También se ve claro que la comunicación con la computadora será cada vez más humana y menos orientada a los especialistas en computadoras y programación. En este sentido se verán más programas en que gran parte de la comunicación se hará a base de tubos de rayos catódicos con "lápices luminosos". Así, el usuario, en lugar de codificar, dibujará su circuito sobre una pantalla de televisión y dará las órdenes a la computadora "apretando botones luminosos". Obtendrá las respuestas en forma gráfica y fácil de interpretar. Apretando un botón luminoso puede ordenar al programa que efectúe dibujos (fig 19), gráficas y tablas finales del diseño. También podrá pedir a la máquina que determine listas de materiales y precios para la fabricación del diseño. Con esto se eliminará mucho trabajo rutinario. Todo lo aquí mencionado ya está en operación en muchas plantas de fabricación de microcircuitos.

Se vislumbra ya que, en un futuro muy cercano, se podrá automatizar desde la concepción del circuito hasta el sistema final. Hasta ahora se han automatizado la mayor parte de los pasos, por ejemplo, el diseño lógico y eléctrico, la documentación y los dibujos, así como la generación de mascarillas para la deposición por métodos ópticos y químicos, y el alambrado

uebs de los circuitos y sistemas. Cuando se logre conquis-

un par de eslabones más, se podrá realizar la completa automatización del proceso. En dicha automatización, la computadora digital desempeña el papel principal.

Como gran parte de los microcircuitos que se fabrican hoy en día son precisamente para la construcción de computadoras, en un futuro cercano será realidad aquel sueño de los escritores de ciencia ficción: máquinas que se reproducen.

Ese sueño hoy en día no es una aberración, pues es cierto, en el sentido estricto de la frase, que las computadoras modernas se diseñan y construyen, en gran parte, ellas mismas.

5. REFERENCIAS

1. G. Kron, *Tensor Analysis of Networks*, John Wiley & Sons, Inc., Nueva York (1939)

2. P. Le Corbeiller, *Matrix Analysis of Electrical Networks*, Harvard University Press, Cambridge, Massachusetts (1950)

3. F. H. Branin Jr., *The Relation Between Kron's Method and the Classical Methods of Network Analysis*, IRE WESCON Convention Record, parte 2 (ago 1959), pp. 3 a 28

4. T. R. Bashkow, *The A Matrix, New Network Concept*, IRE Trans. on Circuit Theory, Vol CT-4 (1957), pp. 117 a 120

5. P. R. Bryant, *The Explicit Form of Bashkow's A-Matrix*, IRE Trans. on Circuit Theory, Vol. CT-9 (1962), pp. 303 a 306

6. F. H. Branin Jr., *DC and Transient Analysis of Networks Using a Digital Computer*, IRE International Convention Record, Vol 10, parte 2 (1962) pp. 236 a 256

7. A. Dervisoglu, *Bashkow's A-Matrix for Active RLC Networks*, IEEE Trans. on Circuit Theory, Vol CT-11 (1964), pp. 404 a 406

8. R. J. Duffin, *Nonlinear Networks*, Boletín American Mathematical Society, Vol 2a (oct 1947), pp. 963 a 971

9. G. J. Minty, *Solving Steady-State Nonlinear Networks of Monotone Elements*, Trans. IRE, Vol 8 (1961), pp. 99 a 104

10. J. Katzenelson, *An Algorithm for Solving Nonlinear Resistive Networks*, Bell System Technical Journal, Vol 44 (nov 1965), pp. 1605 a 1620

11. T. E. Stern, *The Theory of Nonlinear Networks and Systems*, Addison-Wesley Publishing Co., Inc., Reading, Massachusetts (1965)

12. C. A. Desoer y J. Katzenelson, *Nonlinear RLC Networks*, Bell System Technical Journal, Vol 54, No 1 (1965), pp. 161 a 198

13. J. Katzenelson, *AEDNET: A Simulator for Nonlinear Networks*, Proc. IEEE, Vol 54 (nov 1966), pp. 1536 a 1552

14. W. S. Percival, *Solution of Passive Electrical Networks by Means of Mathematical Trees*, Journal of the Institution of Electrical Engineers, Vol 100, parte 2, Londres (1953), pp. 143 a 150

15. W. S. Percival, *Graphs of Active Networks*, Institution of Electrical Engineers, Monografía No 129, Londres (1955)

16. S. J. Mason, *Topological Analysis of Linear Non-Reciprocal Networks*, Proc. IRE, Vol 45 (jun 1957), pp. 829 a 838

17. C. L. Coates, *General Topological Formulas for Linear Network Functions*, Trans. IRE, Vol CT-5 (mar 1958), pp. 30 a 42

18. W. Mayeda y M. E. Van Valkenburg, *Analysis of Non-Reciprocal Networks by Digital Computers*, IRE National Convention Record, Vol 6, parte 2 (1958), pp. 70 a 75

19. W. W. Happ, *Flowgraph Techniques for Closed Systems*, IEEE Trans. on Aerospace and Electronic Systems, Vol 3 (1966), pp. 252 a 264

20. T. J. Kobylarz, *Computer Determination of Symbolic State Equations for Nonlinear Circuits*, Proc. International Conference on Computer-Aided Design, Southampton, Inglaterra (abr 1969), pp. 415 a 425

21. T. Roska, *Generating Network Functions in Literal Form by Digital Computers*, Summer School on Circuit Theory, Praga (1968)

22. J. W. Cooley y J. W. Tukey, *An Algorithm for the Machine Calculation of Complex Fourier Series*, Math. of Computation, Vol 19 (1965), pp. 297 a 301

23. M. A. Murray Lasso, *Analysis of Linear Integrated Circuits by Digital Computer Using Black-Box Techniques*, Chapter 4 in Computer-Aided Integrated Circuit Design, G. J. Herskowitz, McGraw-Hill Book Co., Nueva York (1968)

24. *1620 Electronic Circuit Analysis Program (ECAP), Application Program 1620-EE-02X*, Data Processing Division, IBM Corp., White Plains, Nueva York

25. L. P. McNamee y P. Potash, *A User's Guide and Programmer's Manual for NASAP*, Dept. of Engineering, informe No. 68-38, Universidad de California, Los Angeles (ago 1968)

26. A. F. Malmberg, F. L. Cornwell y F. N. Hofer, *NET-1 Network Analysis Program*, Report LA-3119, Los Alamos Scientific Laboratory, Los Alamos, N. M. (1964)

27. L. D. Milliman, W. A. Massena, R. H. Rickhaut y A. C. Mong, *CIRCUS: A Digital Computer Program for Transient Analysis of Electronic Circuits*, Vol 1 y 2, The Boeing Co., Wáshington (ene 1967)

28. H. W. Mathers, S. R. Sedore y J. R. Senta, *Automated Digital Computer Program for Determining Responses of Electronic Circuits to Transient Nuclear Radiation (SCEPTRE)*, Vol 1 y 2, IBM Technical Report No AFWL-TR-66-126, Air Force Weapons Lab. (feb 1967)

29. G. K. Pritchard, *A Survey of Transient Analysis Programs*, Proc. Computer Aided Circuit Design Seminar, NASA/ERC, Cambridge, Massachusetts (1967)

30. F. F. Kuo, *Network Analysis by Digital Computer*, Proc. IEEE, Vol 54, No 6 (jun 1966), pp. 820 a 829

31. D. E. Meyerhoff y L. P. McNamee, *Considerations for Optimum General Circuit Analysis Program Application*, Proc. Sixth Annual Allerton Conference on Circuit and Systems Theory, Monticell, 3 (oct 1968), pp. 396 a 405

32. H. A. Spang, *A Review of Minimization Techniques for Non-Linear Functions*, SIAM Review, Vol 4 (1962), pp. 343 a 365

33. G. C. Temes y D. A. Calahan, *Computer-Aided Network Optimization: The State of the Art*, Proc. IEEE, Vol 55, No 11 (nov 1967), pp. 1832 a 1863

34. E. B. Kozemchak y M. A. Murray Lasso, *Computer Aided Circuit Design by Singular Imbedding*, Bell System Technical Journal, Vol 48, No 1 (ene 1969), pp. 275 a 315

35. D. T. Ross y J. E. Rodriguez, *Theoretical Foundations for the Computer-Aided Design System*, 1963 Spring Joint Computer Conference, Proc. AFIPS, Vol 23, Spartan Books, Inc., Wáshington (1963)

36. D. S. Evans, y J. Katzenelson, *Data Structure and Man-Machine Communication for Network Problems*, Proc. IEEE, Vol 55, No 7 (jul 1967), pp. 1135 a 1144

37. H. C. So, *Analysis and Iterative Design of Networks Using On-Line Simulation*, en la ref 38

38. F. F. Kuo y J. F. Kaiser, *System Analysis by Digital Computer*, John Wiley & Sons, Inc., Nueva York (1966)

39. G. J. Herskowitz, *Computer-Aided Integrated Circuit Design*, McGraw-Hill Book Co., Nueva York (1968)

40. D. A. Calahan, *Computer-Aided Network Design*, edición preliminar, McGraw-Hill Book Co., Nueva York (1968)

41. R. W. Jensen y M. D. Lieberman, *IBM Electronics Circuit Analysis Program: Techniques and Applications*, Prentice-Hall Inc., Englewood Cliffs, Nueva Jersey (1968)

42. G. W. Zobrist, *Network Computer Analysis - ECAP, NASAP, NET-1, SCEPTRE and Modeling*, Boston Technical Publishers, Inc., Cambridge, Massachusetts (1968)

43. *Special Issue on Computer-Aided Design*, Proceedings IEEE, Vol 55, No 11 (nov 1967)

44. *Special Issue on Computer Oriented Microwave Practices*, IEEE Transactions on Microwave Theory and Techniques, Vol MTT-17, No 8 (ago 1969)

45. *Special Issue on Educational Aspects of Circuit Design by Computer*, IEEE Transactions on Education, Vol E-12, parte 1, No 3 (sept 1969), parte 2, No 4 (dic 1969)

46. *Circuit Design By Computer Symposium*, Tutorial Symposium, Universidad de Nueva York, Nueva York (ene 31, feb 2, 1967)

47. *Computer-Aided Circuit Design Seminar*, NASA-Electronics Research Center, Cambridge, Massachusetts (abr 1967)

48. *Conferencia sobre análisis de circuitos con computadora digital*, UNAM, México (jun 1967)

49. *Conferencia sobre análisis de circuitos con computadora digital*, Instituto Tecnológico y de Estudios Superiores, Monterrey, México (jul 4 y 5, 1967)

50. P. R. Mayaguez, *Conferencia sobre el uso de las computadoras en el diseño eléctrico*, Universidad de Puerto Rico (sep 1 7)

51. *Computer-Aided Integrated Circuit Design Course*, Instituto Tecnológico de Stevens (sep 11 a 15, 1967)

52. *Automated Circuit Analysis Course*, Universidad de California, Los Angeles (abr 3 a 7, 1967)

53. *Computer-Aided Design Workshop*, Universidad de Missouri, Columbia, Missouri (ago 7 a 18, 1967)

54. *Computer Oriented Circuit Design: A Simbolic Approach*, Universidad de California, Los Angeles (ene 22 a 26, 1968)

55. *Design Automation Workshops*, Annual Meetings on Computer Aided Design, ACM, SHARE y IEEE

56. *Computer Aids for Large Circuit Arrays*, Universidad de sconsin, Madison, Wisconsin (sep 27 a 28, 1967)

57. *Computer-Aided Circuit Design*, A Compiler Oriented Approach, Universidad de California, Los Angeles (feb 3 a 7, 1969)

58. *International Conference on Computer Aided Design*, IEEE (Londres) Southampton, Inglaterra (abr 15 a 18, 1969)

59. *Computer Aided Integrated Circuit Design*, IEEE New Technical and Scientific Activities Committee, Hoboken, Nueva Jersey (jun 5, 1967)

60. *Annual Summer Institute on Computer-Aided Circuit Analysis and Design*, Universidad de Missouri, Columbia, Missouri (ago 5 a 16, 1968)

61. *Computerized Electronics*, Universidad Cornell, Ithaca, Nueva York (ago 26 a 28, 1969)

62 *CODING INSTRUCTIONS FOR NASAP 69/I (Network Analysis for Systems Applications Program)* Revision No 1, Gaertner Research Incorporated, Stamford, Connecticut (ene 8, 1969)

63. E. A. Guillemin, *Theory of Linear Physical Systems*, John Wiley and Sons, Inc., Nueva York (1963)

64. M. A. Murray Lasso, *Black-Box Models for Linear Integrated Circuits*, IEEE Transactions on Education, Vol E-12, No 3 (sep 1969), pp. 170 a 180

65. W. W. Happ, *Combinatorial Analysis of Multi-terminal Devices*, IEEE Transactions on Systems Science and Cybernetics, Vol SSC-3, No 1 (jun 1967), pp. 21 a 27

66. T. R. Nisbit y W. W. Happ, *The Calculus of Deviations Applied to Transistor and Network Analysis*, J. British IRE, Vol 21 (1961), pp. 437 a 450

67. M. A. Murray Lasso y E. B. Kozemchak, *Microwave Circuit Design by Digital Computer*, IEEE Transactions on Microwave Theory and Techniques, Vol MTT-17, No 8 (ago 1969), pp. 514 a 526

68. M. A. Murray Lasso y E. B. Kozemchak, *Foundations of Computer-Aided Design by Singular Imbedding*, IEEE Transactions on Circuit Theory, Vol CT-17, No 1 (feb 1970), pp. 105 a 112

MICROWAVE CIRCUIT DESIGN BY DIGITAL COMPUTER

BY

M. A. MURRAY-LASSO AND E. B. KOZEMCHAK

Reprinted from IEEE TRANSACTIONS
ON *MICROWAVE THEORY AND TECHNIQUES*
Volume MTT-17, Number 8, August, 1969
pp. 514-516

Microwave Circuit Design by Digital Computer

MARCO A. MURRAY-LASSO, MEMBER, IEEE, AND EDWARD B. KOZEMCHAK

Abstract—Methods for the automatic analysis and design of microwave circuits using a digital computer in batch mode are given. The methods are capable of handling microwave components modeled by ordinary R, L, C, M, CS elements plus transmission lines and multiterminal black-boxes whose characteristics have been determined theoretically or experimentally. The analysis-optimization program, IMPROVE (Integrated and Microwave PRogram for Optimizing Variable Elements), implementing the methods presented in this paper is described its use illustrated with a practical design problem.

I. Introduction

CONSIDERABLE progress has been made in the field of analysis of electronic circuits by digital computer in the last five years. Presently, several general purpose computer programs for dc, ac, and transient analysis of linear and nonlinear circuits are widely available [1]–[6]. The design of a circuit is a trial and error process generally accomplished by repeated analysis and parameter or structure changes done by the designer until the response of a circuit is sufficiently close to the desired response. This requires several approaches to the computer and careful evaluation of the results of each run. The procedure can be very time consuming.

A portion of this process can be automated so that the computer does the evaluation and parameter changes and the designer decides only on the structure and the values of a set of fixed parameters. Several researchers have reported various degrees of success in applying these techniques to lumped circuits [7]–[12]. The automatic parameter changes are accomplished by converting the design problem into the problem of minimizing a function of several variables subject to a set of inequality constraints. To this problem a number of mathematical techniques may be applied [13]–[15]. By automating the analysis, evaluation, and parameter changes portion of the design process, the time to complete a design can be shortened considerably.

The methods mentioned above have not been widely applied to microwave circuits due to the inability of the analysis programs to conveniently handle distributed circuits or circuits that have been characterized empirically. Hence, automatic circuit design has been largely restricted to low frequency applications.

In this paper some methods appropriate for microwave circuits are presented and a program implementing them is

Manuscript received March 10, 1969; revised May 8, 1969. This work was partially supported by NASA Electronics Research Center under Grant NAS 12-2107.

M. A. Murray-Lasso is with Systems Research Center, Case Western Reserve University, Cleveland, Ohio.

E. B. Kozemchak is with Bell Telephone Laboratories, Whippany, N. J.

described. IMPROVE (Integrated and Microwave PRogram for Optimizing Variable Elements) is a batch program capable of automatically designing integrated and microwave networks in the frequency domain. It couples a general purpose analysis program BELNAP [16] with a direct pattern search optimization scheme. The user inputs a network description specifying variable parameters and their limits plus a desired performance. The optimization seeks new parameter values to minimize a weighted mean-squared error criterion. The output of the program is a final set of parameter values which produces a performance as "close as possible" to the desired one. The use of the program is illustrated with an example.

II. Automatic Design

For some design requirements synthesis procedures exist to determine both topology and parameter values [17]–[22]. For many requirements, however, no straightforward synthesis exists; this is true particularly in networks containing active devices at sufficiently high frequencies so that simple low frequency models give poor approximations to real behavior. For these cases the designer's experience and intuition are brought to bear in choosing a circuit to meet the requirements. A cut and try procedure is usually adopted in order to determine the appropriate parameter values to meet the requirements.

The design process using the cut and try scheme is indicated in Fig. 1. The operations done best by man are marked M, those done best by computer are marked C and those done by either or both combined are marked MC. The problem is created when a system need is recognized and defined. (For example, it is decided to amplify signals in a specified frequency band.) The need is then expressed as a circuit. (It is decided to build a 20 dB narrow-band amplifier.) Next the objectives are expressed in circuit terms. (The transducer gain must be a constant of value 10 over the band of interest; the input impedance must be 50 ohms in the band of interest.) Next a candidate schematic is chosen. (Active elements for amplification are selected, circuitry for biasing the active elements and for matching the input impedance is added. Elements for producing a flat maximum of sufficient magnitude at the center frequency of interest are added.) An initial parameter guess is made. (Actual values for the parameters of both the active and passive elements are inserted. This may involve making educated guesses about parasitics or coming up with parameters such as lengths and position of stubs from simplified calculations.) An analysis of the circuit is made with the aid of an analysis program. The behavior of the circuit is compared with that of an ideal circuit meeting all the objectives. (The transducer gain and input impedance are calculated for several frequencies in the band of interest and compared with the values specified.) If the circuit meets the objective the design is finished (quite unlikely on the first trial) and a breadboard is built. If the objectives are not met and a preset maximum number of trials or schematics has not been exceeded, the parameter values are changed by the designer using as much experience and

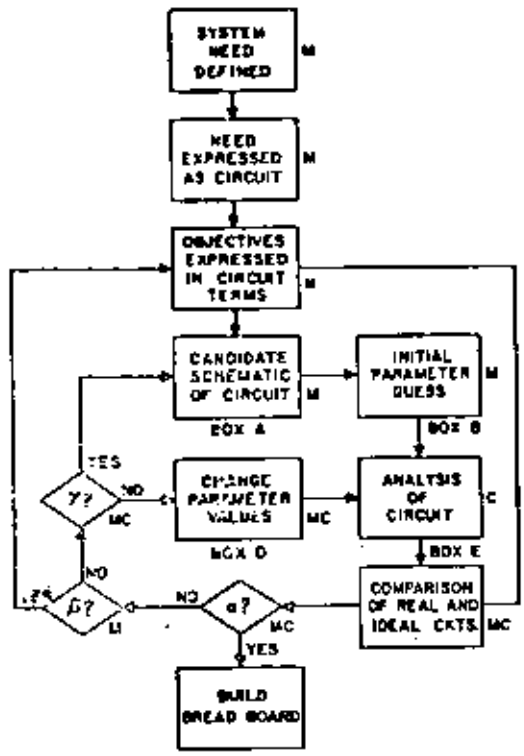


Fig. 1. Simplified diagram of the computer-aided design process

intuition as possible. (For example, the parameters may be positions and lengths of stubs.) The change should reduce the difference between the actual performance and the desired performance. (For example, if the gain is low for some frequencies, the change should raise it.) A new analysis of the circuit reveals that indeed the new set of parameters does give improved performance. The loop "analysis of circuit—comparison of real and ideal circuits—change parameter values" is traversed many times until either the objectives are met with sufficient approximation or too many trials have been attempted. If the latter is the case the designer may try a new candidate circuit and repeat the process. It may happen that the number of schematics tried unsuccessfully exceeds a preset maximum. The designer then uses the outermost loop and changes the circuit objectives (for example, reduces the gain requirement). Additional loops could be incorporated but are not shown in Fig. 1 for simplicity.

In many problems the effect of parameter values on performance is complicated and the designer lacks intuition in determining parameter changes to improve performance. In such cases exploratory changes of parameters can be made to determine appropriate changes. This is feasible if each analysis is done with the aid of the computer. It is not difficult, however, to run into cases where the process becomes very laborious if several parameters are varied. Furthermore each analysis requires the user to look at printouts or plots, an operation which takes time and effort, especially if the design is done using the batch processing method. (This item

is one of the strong motivations for the introduction of time-sharing in computer aided design.)

By defining an appropriate numerical criterion of performance and a search strategy for the parameter value ranges, the innermost loop of Fig. 1 may be completely automated. Both operations are simple to program, but the eventual overall success of the method depends strongly upon how it is done. This is due to the fact that although qualitatively almost any search strategy applied to any initial set of parameters will converge to some local minimum error between the desired response and the actual response, quantitatively (in computer time) different search techniques will converge at very different rates. This point will be discussed at greater length in Part IV.

We now proceed to concentrate on the analysis box of Fig. 1 insofar as microwave circuits are concerned.

III. Computer Analysis of Microwave Circuits [16]

The main trouble with the circuit analysis programs available is that they suffer from one of the following defects.

1) They do not handle distributed circuits.
2) They do not accept circuits characterized empirically.
3) They do not handle general configurations.
4) They do not calculate quantities of interest to microwave engineers.

A circuit analysis program, BELNAP [16], designed specifically to avoid the problems mentioned above was implemented and is used at Bell Telephone Laboratories by many circuit designers working with microwaves and integrated circuits. The operation of BELNAP is very simple. It is based on well-known tools of multiport circuit analysis. With relatively little effort any nodal analysis program can be modified to avoid the problems mentioned above. A short review of the basic ideas is now given. Additional details may be obtained from [16].

All elements in a circuit are considered to be multiterminal black-boxes by BELNAP. Each one of these black-boxes is characterized by an indefinite admittance matrix (IAM), which is generally frequency dependent. The frequency dependency of the block boxes may be given to the computer in several ways, namely by:

1) analytical expressions,
2) tables of discrete values,
3) parameters in interpolating or approximating expansions.

For example, a capacitor (which may represent a fringing electric field) connected between nodes 2 and 4 in an 8 node circuit has an IAM which is an 8×8 matrix all of whose entries are zero except

$$Y_{22} = Y_{44} = Cj\omega, \qquad Y_{24} = Y_{42} = -Cj\omega. \tag{1}$$

In this case analytical expressions give the frequency dependency of the IAM of the capacitor. Another case arising in microwave circuits of an IAM given by analytical expressions is the IAM of a pair of equal coupled lines of parameters per unit length, L_{11}, L_{12}, C, C_M and length l connected

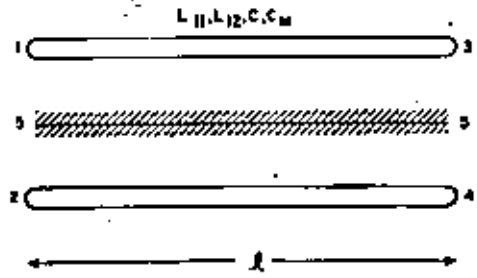


Fig. 2. Pair of equal coupled-lines of length l over a ground plane. Quantities L_{11}, L_{12}, C, C_M are per unit length.

to terminals 1, 2, 3, 4 as shown in Fig. 2. If this pair of lines is imbedded in a circuit with 8 terminals, the corresponding IAM is an 8×8 matrix all of whose entries are zero except the following:

$$\begin{aligned}
&Y_{11} = Y_{22} = Y_{33} = Y_{44} = \tfrac{1}{2}A \coth (Ul) + \tfrac{1}{2}B \coth (Vl)\\
&Y_{12} = Y_{21} = Y_{34} = Y_{43} = \tfrac{1}{2}A \coth (Ul) - \tfrac{1}{2}B \coth (Vl)\\
&Y_{13} = Y_{24} = Y_{31} = Y_{42} = -\tfrac{1}{2}A \operatorname{csch} (Ul) - \tfrac{1}{2}B \operatorname{csch} (Vl)\\
&Y_{14} = Y_{23} = Y_{41} = Y_{32} = -\tfrac{1}{2}A \operatorname{csch} (Ul) + \tfrac{1}{2}B \operatorname{csch} (Vl) \qquad (2)\\
&Y_{16} = Y_{25} = Y_{38} = Y_{47} = Y_{51} = Y_{62} = Y_{73} = Y_{84}\\
&\quad = A[\operatorname{csch} (Ul) - \coth (Ul)]\\
&Y_{55} = \tfrac{1}{2}A[(\coth (Ul) - \operatorname{csch} (Ul)]
\end{aligned}$$

where

$$A = \sqrt{\frac{C}{L_{11} + L_{12}}}, \qquad B = \sqrt{\frac{C + 2C_M}{L_{11} - L_{12}}},$$

$$U = j\omega\sqrt{(L_{11} + L_{12})C}, \qquad V = j\omega\sqrt{(L_{11} - L_{12})(C + 2C_M)}.$$

Similar expressions may be obtained for single transmission lines, RC lines and certain waveguides, etc., and incorporated into an analysis program.

Many microwave elements are difficult to model analytically because of the difficulty in solving the corresponding fields in closed form. This difficulty occurs either because not enough is known about the device or because of imperfections in manufacturing. Most of the active devices and those with fringing fields (bends, discontinuities, cavities, etc.) fall into this category. These may be characterized by numerical calculations of the fields or by external measurements of their scatterings at discrete frequencies. By mathematical transformations the IAM matrix can be obtained numerically at discrete frequencies. For example, if the scattering matrix of a multiport is measured at a given frequency, with all ports having a common terminal, the (definite) nodal admittance matrix Y is obtained by the equation [23]

$$Y = R_0^{-1/2}(I + S)^{-1}(I - S)R_0^{-1/2} \tag{3}$$

where I is the unit matrix and R_0 is a diagonal matrix whose entries are the terminating loads at the ports during the measurements. The IAM is then obtained from Y by adding a row and column equal to the negative of the sums of the rows and columns of $-Y$. This is done at each measured frequency.

Intermediate frequency values may be obtained by any

method of approximation or interpolation [24]. Linear interpolation has been found to give adequate results for most cases. It is now the standard method used by BELNAP with polynomials in frequency, polynomials in the log of frequency and log-log graph polynomials available on request. To avoid accuracy problems, orthogonal polynomials are used. The orthogonal polynomials used are special sets which have been orthonormalized over an adjustable frequency interval in multiple precision by the Gram-Schmidt method [24]. The different sets of polynomials were chosen so that some curve shapes which appear often are well approximated with three or four terms. More sophisticated methods such as splines [25] could be used with more aesthetic results but with correspondingly more complicated computations. When a table has been approximated by polynomials only the coefficients need to be stored. Once each subcircuit (two terminal elements being particular cases) of a circuit is characterized through its IAM as a black-box according to any of the three methods mentioned, the IAM of the complete circuit is found by simply adding the corresponding IAMs.

At any point in the analysis internal terminals may be eliminated and the IAM at specified terminals of the complete circuit or a subcircuit found numerically at discrete frequencies. The mathematics of this operation are quite simple. Let I_1 and I_2 be vectors of currents entering the p external terminals and g internal terminals, V_1 and V_2 the corresponding vectors of voltages (with a floating node as reference), and Y_{11}, Y_{12}, Y_{21}, Y_{22} submatrices of the IAM of proper dimensions for the equation

$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \tag{4}$$

to be properly partitioned. Elimination of V_2 yields

$$I_{eq} = Y_{eq} V_1 \tag{5}$$

where

$$I_{eq} = I_1 - Y_{12} Y_{22}^{-1} I_2, \quad Y_{eq} = Y_{11} - Y_{12} Y_{22}^{-1} Y_{21}. \tag{6}$$

I_{eq} is a vector of equivalent independent sources connected to the external terminals to account for the internal sources. Y_{eq} is the IAM of the network with the internal nodes suppressed. If no internal sources exist then $I_2 = 0$ and $I_{eq} = I_1$.

IV. Automatic Parameter Optimization

To automate the inner loop of Fig. 1 once an analysis program is available, it is necessary to

1) define a desired response in numerical terms,
2) define a criterion of performance or "distance" between the desired response and actual response,
3) define an optimization or minimization strategy.

The definition of a desired response normally means that the user specifies quantities like return loss, insertion gain or voltage standing wave ratio which he desires the circuit to achieve at several discrete frequencies. The specified quantities may be thought of as a point in N-dimensional space. (For example if the return loss magnitude at four frequencies is specified and the value of the return loss magnitude at each frequency is measured on an axis, a circuit response is a 4-dimensional point. The "ideal circuit" response is also a 4-dimensional point in the same space.) A distance is then defined in the space. One of the most popular distances is the so called "Euclidean distance" defined as the square root of the sums of the squares of the differences between the coordinates of the ideal circuit response and the actual circuit response. Other definitions are possible, for example instead of squares, absolute values may be used. The Chebyshev norm (maximum deviation) and pth norms have been found to produce terrain which is more rugged and hence more difficult to explore than the Euclidean norm. They are also more difficult to compute. For many cases it is convenient to "weigh" some frequencies more than others. IMPROVE uses a double weighted Euclidean distance as will be explained below.

It is not necessary that the point defined by the ideal circuit response correspond to only one quantity or only one port. The ideal circuit response may involve quite complicated conditions, for example, return loss at ports one and four and insertion loss at ports one and two, one and three and one and four; all at five discrete frequencies. It may be more important to achieve the right return loss than the insertion losses, hence one may weigh return losses with five and insertion losses with unity. Furthermore, one may weight the center frequency more than the frequencies at the edges of the band.

Finally, it is not necessary that the ideal circuit response be defined as a point. In some cases one would like to define it as a region. For example for a notch filter one may be satisfied to have at least 50 dB of loss at a given frequency rather than exactly 50 dB. Since such types of specifications are easily implemented with logical decision statements on the computer, they should be considered in designing a program.

The definition of desired response and criterion of performance are very delicate matters because the final results will heavily depend on them. In order to come up with good definitions the designer is forced to ask himself "What do I really want the circuit to do?" This question is a difficult one. A designer would usually prefer to decide a posteriori which of several circuits he likes most rather than to have to state a priori the characteristics of the best one. Not only will the final results differ for different definitions of desired response and criterion of performance, but also, the time for a search strategy to arrive at them. For example it is well-known by researchers in automatic optimization that a "Chebyshev distance" (the distance between two curves being given by their maximum deviation) is apt to present a much more rugged terrain (if an analogy between the function to be minimized and height in a surveyor's map is made) than Euclidean distance. The rugged terrain will force the search strategy to use large quantities of computer time. Another typical example is when a naive designer specifies a bandpass filter with sharp corners and vertical walls. Knowing that he cannot expect sharp corners he should round them in a reasonable manner, otherwise the computer will

spend all its time trying to square the corners and trying to achieve the vertical walls at the expense of producing undesirable bulges in the band of interest. The end result is not all what the designer expected. Usually it gives a horrible response by anyone's standards. This could be corrected by proper frequency weighing. The only way to pick up all these pieces of information is by experience with different types of circuits and different definitions of desired response and criterion of performance. This is an area where interactive computing has a great application.

In implementing the search strategy one has quite a bit of choice [7], from grid search to gradient techniques of all kinds to random search. Each strategy has its own advantages and disadvantages and its own requirements. For example, gradient techniques require partial derivative information which is very hard to come by for complicated microwave circuits composed of combinations of analytically and experimentally characterized subnetworks. One possible way of obtaining partial derivatives is to find them numerically by evaluating the response with each parameter incremented a small amount. This is usually expensive computationally. Because of the problems in obtaining partial derivatives for microwave networks plus the fact that the terrains are usually rugged enough not to be worth the sophistication of "differentially" following the surfaces, it was decided to use a direct pattern search technique for IMPROVE. The one finally implemented was proposed by Hooke and Jeeves [26]. The direct search method has a historic record of successfully climbing in very rugged terrain (which is usually the case in circuit problems), and growing linearly with the number of parameters. It is also quite easy to handle constraints on the values of the parameters. These are necessary to end with meaningful engineering results. (For example, it would be very undesirable to end with a negative characteristic impedance for a transmission line.)

The direct pattern search strategy has two major components: the exploratory move and the pattern move. Briefly, the strategy is described as follows. An initial guess is made of the n parameters for a problem and some error criterion is evaluated at that set of parameter values with the aid of an analysis program. Starting with the initial set of parameters, an exploratory move is made. This exploratory move varies the value of each parameter by some small amount and observes the effect of each of these variations on the error expression. Those changes that reduce the error are retained. It may be that certain of the parameters are slightly increased, others decreased, and still others unchanged to reduce the error. This new set of slightly modified parameters determines the "unit vector" in n-dimensional parameter space of a move which will reduce the error. The next step is to make a larger move in the parameter values in the direction indicated by the exploratory move. This larger move is called a pattern move. After the pattern move has been made, the error expression is again evaluated for the new parameter values. If the pattern move succeeded in reducing error, another pattern move is made in the same direction as the first one. The pattern moves continue until one fails to reduce the error. At this point, a new exploratory move is made to determine a new direction for a pattern move. This entire process continues until an exploratory move is unable to reduce the error expression further. At this point, the program terminates. The flow chart of Fig. 3 outlines the pattern search strategy. The details of the exploratory move are indicated in Fig. 4. The steps are repeated for each parameter of the system.

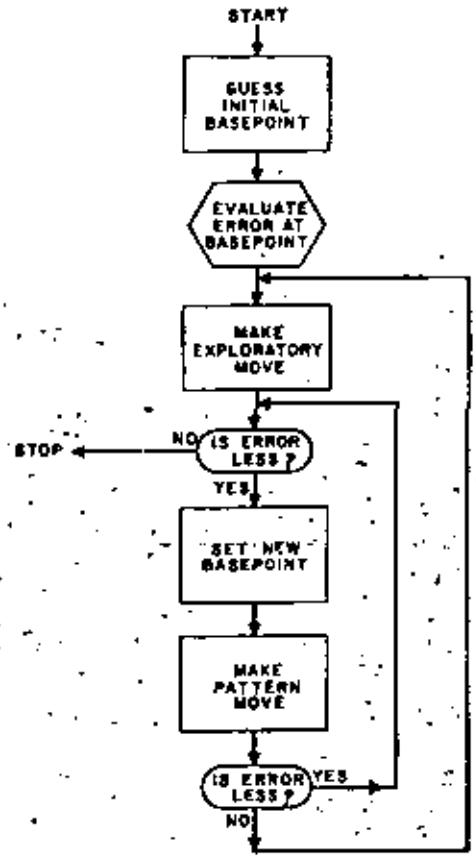


Fig. 3. Flowchart of the pattern search strategy.

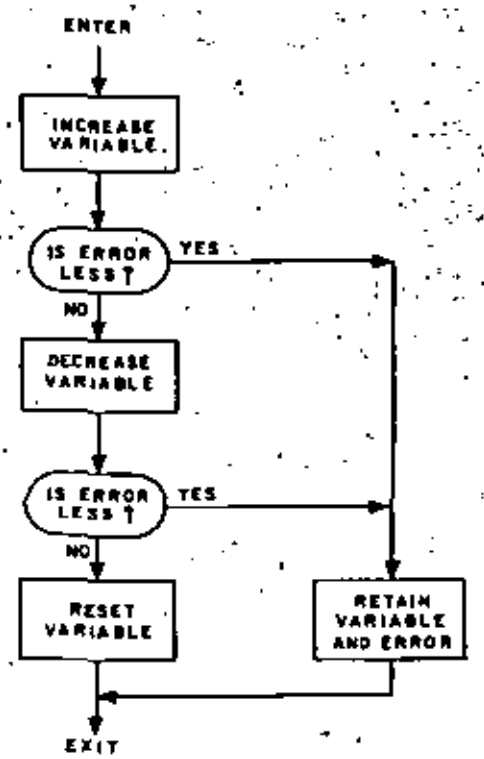


Fig. 4. Flowchart of the exploratory search strategy.

V. Improving the Computational Efficiency Through the IAM Black-Box Approach

A feature deemed critical in coupling an analysis program to an optimization routine is the numerical efficiency of each analysis. In a typical optimization run, the analysis program may be called several hundred times, therefore any saving in the computational analysis effort is multiplied by several hundred.

The indefinite admittance matrix formulation employed by IMPROVE allows a significant decrease in computational effort. In the optimization some subcircuits have no adjustable parameters. Since this fixed part of the network does not change with different parameter set trials, it should not be necessary to solve the equations of the invariant portion for every new parameter set. To avoid the repetitious analysis of the fixed part, the network is partitioned into its variant and invariant portion. The fixed portion of the network is characterized at its essential terminals and stored as a black-box. Essential terminals are those which are specified as external ports or those which have adjustable elements connected (see Fig. 5). To save computer core storage the subcircuits can be reduced to the essential terminals in preliminary passes. This can easily be done because BELNAP can punch on request the IAM of a circuit reduced to a given set of terminals. The punched data is in a format in which it can be read for a subsequent analysis or optimization. In this way very large circuits containing few variable elements can be handled in several passes.

In order to characterize the fixed portion of the network at its essential terminals, the nonessential terminals are eliminated by using (5) and (6). H. C. So [27] uses this method with a hybrid formulation for lumped circuits and reports considerable computer-time savings when compared with efficient programs not using the partitioning technique. It has not been possible to compare IMPROVE in microwave problems with other analysis programs because of the unavailability of other general configuration programs capable of handling distributed multiport networks. However, we expect that the relative savings from the IAM black-box approach would be even greater than those reported by So, because he has to consider the time necessary for the computation of the hybrid matrix at each frequency. In our approach, computation of an IAM involves only adding the IAMs of the subcircuits. Additionally, the algorithms for hybrid analysis are considerably more complicated than those for nodal analysis so the programming effort for implementing our method is considerably smaller.

The principal reason given by So [27] for using general hybrid analysis is that, since the adjustable parameters may be distributed arbitrarily in the network, the open-circuit impedance matrix or short-circuit admittance matrix of the resulting n-port frequently does not exist. This is particularly true of open-circuit impedance matrices. In practical circuits the authors have yet to find a single case in which the IAM of any subcircuit imbedded in a real network does not exist. Granted that networks such as ideal transformers and perfectly coupled mutual inductances do not have admittance or impedance matrices. These devices are, however,

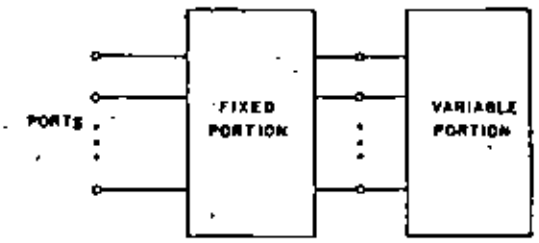


Fig. 5. Network partitioned into its invariant and variable parts for analysis by IMPROVE.

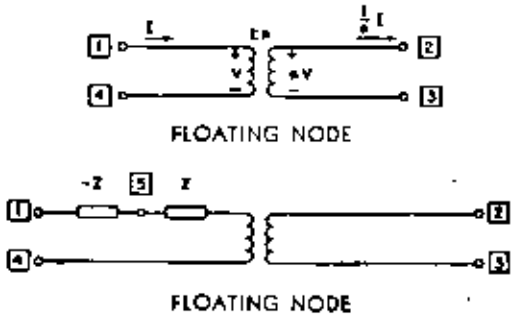


Fig. 6. Ideal transformer has no indefinite admittance matrix. Addition of virtual node 5 removes degeneracy.

theoretical idealizations which never appear in microwave networks. On the other hand, even from a purely theoretical point of view, as long as the complete circuit possesses an IAM, the problem can easily be circumvented as follows: [16] for every degenerate device (such as an ideal transformer) insert a virtual node (a node which does not exist physically) to which positive and negative impedances of equal values are connected. These cancel each other's effects and therefore leave the circuit undisturbed. This is shown in Fig. 6 for an ideal transformer. The additional virtual node destroys the degeneracy because the circuit with an additional node does have an IAM. The virtual node is eliminated after the additional subcircuits are connected. This occurs when the total IAM at the real terminals is no longer degenerate. A computer program can accept positive or negative parameters with equal ease. Since the value of the canceling artificial impedances can be chosen by the designer, they may be chosen so that they are comparable to other impedances in the network. That is, if the other impedances are of the order of 10^6 ohms the artificial impedances should also be of the same order. This simple artifice also has application in other instances, such as obtaining Norton equivalents of voltage sources without a series impedance and avoiding ill conditioned equations when impedances of very different magnitudes are connected to the same node [28].

VI. Brief Description of IMPROVE (Example)

The method of analysis used by IMPROVE is identical to that of the program BELNAP [16]. The circuits handled may contain:

1) Passive R's, L's, and C's.
2) Uniformly distributed transmission lines through the specification of R, L, G, and C per unit length and the

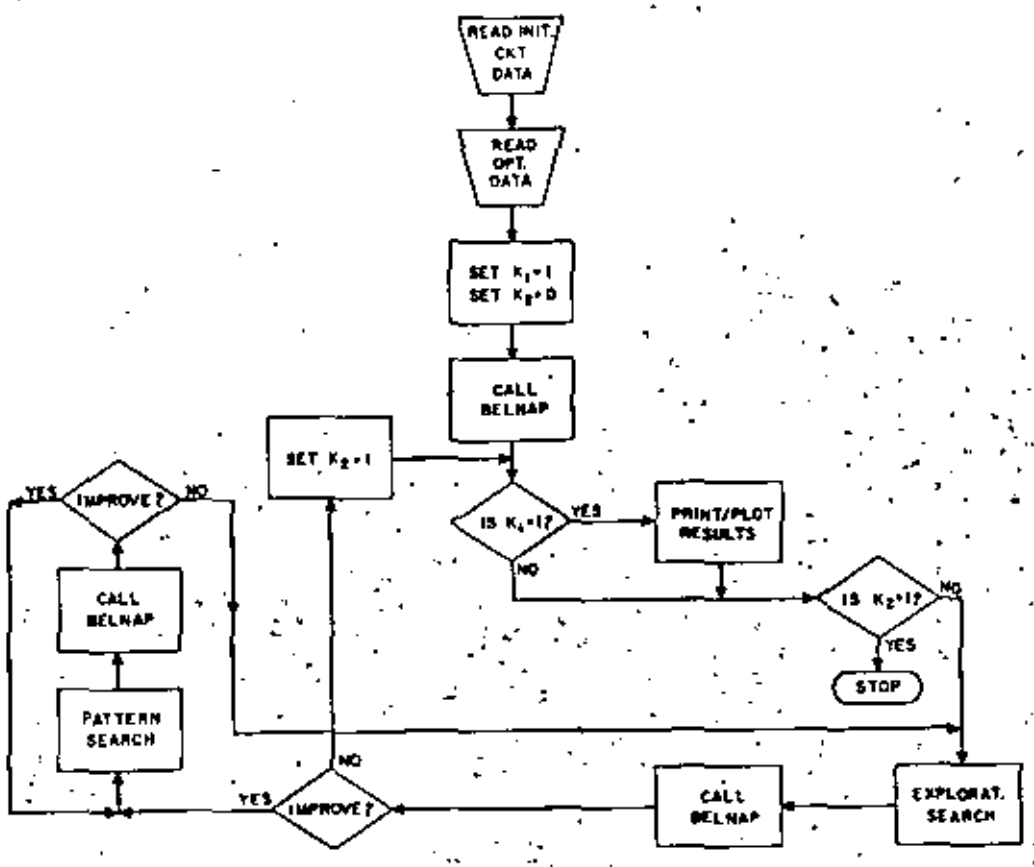


Fig. 7. Simplified flowchart of analysis-optimization program IMPROVE.

length or characteristic impedance and electrical length at a given reference frequency.

3) Active devices for which conventional modeling with controlled sources and passive elements is adequate.
4) "Black-boxes" for which conventional modeling is difficult or impossible but whose terminal characteristics (S, Y, Z, H, or $ABCD$ parameters) can be obtained. (For example, one can make terminal measurements at several frequencies on a microwave transistor, a directional coupler or similar black-box and input this data without first modeling the device. Intermediate frequency values are interpolated by the program.) Theoretically modeled devices may be defined using external subroutines.

Additionally for optimization purposes IMPROVE accepts the following specifications from the user:

1) Variable parameters (including minimum and maximum values for all variable parameters). Parameters which vary together.
2) Frequencies and weighting factors for the frequencies.
3) Desired performance characteristics.
   a) Impedance (magnitude and/or angle or real and/or imaginary parts).
   b) Return loss.
   c) Insertion gain (magnitude in dB or phase).
   d) Voltage gain (real, imaginary, magnitude and/or angle).

   Any of the previous up to 4 ports.
4) Weights for the performance characteristics. (For example, a user may weigh impedances 10 and insertion gains 1.)

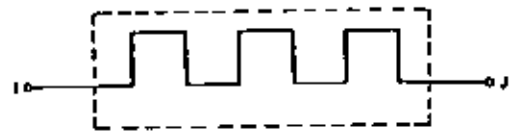
The parameters which may be declared variable in IMPROVE are resistors, capacitors, inductors, controlled sources, per-unit-length parameters of transmission lines (R, L, C, G), lengths of lines, characteristic impedances of lines, and electrical lengths of lines. Because some parameters may vary together it is possible to approximate some distributed elements with lumped circuits and thus determine their per-unit-length parameters approximately. Also it is possible to force some circuits to retain certain desirable physical symmetries.

A simplified flow chart of the operation of IMPROVE is shown in Fig. 7.

The program outputs the value of the error between the desired response and the response of the first and final trial circuits, a list of the final circuit parameters and a complete analysis (including plots) of the final circuit.

Two examples are now presented to illustrate the use of IMPROVE and BELNAP. One of the biggest problems facing microwave engineers is that of unwanted parasitics and their modeling to predict correct circuit behavior. This modeling can be formulated as an optimization or design problem. When the modeling of parasitics is solved, the parasitics can be used in favor of the designer to fine-tune circuits. This is especially important when many circuits will be mass produced.

Consider now the 20 dB switched microwave attenuator of Fig. 9. We first wish to characterize each of the elements



INDEFINITE MATRIX OF CHOKE
ALL ENTRIES ARE ZERO, EXCEPT
$Y_{11} = Y_{22} = -Y_{12} = -Y_{21}$

FREQUENCY	REAL PART	IMAGINARY PART
.9 x 10^9	3.31 x 10^-6	-1.98 x 10^-3
1.04 x 10^9	3.13 x 10^-6	-1.55 x 10^-3
1.17 x 10^9	2.9 x 10^-6	-7.58 x 10^-4
1.3 x 10^9	2.8 x 10^-6	-1.55 x 10^-4
1.43 x 10^9	2.9 x 10^-6	7.55 x 10^-4
1.56 x 10^9	3.12 x 10^-6	1.55 x 10^-3

Fig. 8. Choke characterized as black-box for analysis with BELNAP.

within the circuit and use the BELNAP program to analyze the performance of the ideally (no parasitics) interconnected network.

The diodes were modeled by equivalent circuits as shown in Fig. 9. The values of the elements in the model are conveniently determined by using the optimization program IMPROVE to match measured data on the diodes. This approach was used here to illustrate equivalent circuit modeling. Alternatively, measured data on the device may be directly fed into the program as in the case of the quarter wavelength chokes of the circuit which are shown in Fig. 8 as two-terminal black-boxes. Here, measured data on the Y parameters of the chokes are used as a description to the program. As shown in the input description of Table I, the choke is given as type number (10) and the Y parameters given at 6 frequencies (0.9 GHz through 1.56 GHz). Also shown in Table I is the input of Rs, Ls, and Cs, described by their connection nodes and element values.[1] The three transmission lines are incorporated into the circuit by giving their connection nodes, R, L, G, C per unit length, and length of the line.

The configuration of Fig. 9 was built and its performance measured. The measured values of return loss and insertion gain are shown in Table II. Observe that they differ noticeably from the response predicted by the computer (shown under initial value). The discrepancy can be attributed to parasitic effects in the final circuit layout. These parasitics include fringing capacitance at the termination of each of the transmission lines (including input and output), lead inductance associated with each of the diodes (except diode 1 which required virtually no lead length in the final layout) and inductance associated with each of the resistors.

It was desirable to determine the value of the parasitic elements which were accounting for the discrepancy between predicted and measured response. To do this, the measured data was used as direct input to the optimization program.

[1] Note in Table I the presence of several capacitors with values 10^{-20} and inductors of 10^{-12}. These were introduced here to avoid having to recode the circuit for the optimization run. The inputs of BELNAP and IMPROVE are compatible.

TABLE I

COOLED INPUT TO BELNAP OF THE CIRCUIT OF FIG. 12

```
TWENTY DB SWITCHED MICROWAVE ATTENUATOR
  $INPUT NODES   20, PORTS 2, FREQI 1.E9, FREQF 1.5E9,
    FRQINC .05E9, PLOT 1 $
  $BRANCH NODES 3,0, C .3E-12 $
  $BRANCH NODES 6,0, C .2E-12 $
  $BRANCH NODES 3,6, L .3E-9 $
  $BRANCH NODES 6,0, R .5 $
  $BRANCH NODES 3,4, C .3E-12 $
  $BRANCH NODES 3,18, C .2E-12 $
  $BRANCH NODES 4,18, L .3E-9 $
  $BRANCH NODES 3,18, R 2500. $
  $BRANCH NODES 7,9, C .3E-12 $
  $BRANCH NODES 8,9, C .2E-12 $
  $BRANCH NODES 7,8, L .3E-9 $
  $BRANCH NODES 8,9, R .5 $
  $BRANCH NODES 2,13, C .3E-12 $
  $BRANCH NODES 13,14, C .2E-12 $
  $BRANCH NODES 2,14, L .3E-9 $
  $BRANCH NODES 13,14, R .5 $
  $BRANCH NODES 2,15, C .3E-12 $
  $BRANCH NODES 2,17, C .2E-12 $
  $BRANCH NODES 16,17, L .3E-9 $
  $BRANCH NODES 2,17, R 2500. $
  $BRANCH NODES 19,0, R 61.1 $
  $BRANCH NODES 20,0, R 61.1 $
  $BRANCH NODES 5,11, R 247.5 $
  $BRANCH NODES 1,0, C 1.E-20 $
  $BRANCH NODES 2,0, C 1.E-20 $
  $BRANCH NODES 3,0, C 1.E-20 $
  $BRANCH NODES 15,0, C 1.E-20 $
  $BRANCH NODES 9,0, C 1.E-20 $
  $BRANCH NODES 10,0, C 1.E-20 $
  $BRANCH NODES 11,0, C 1.E-20 $
  $BRANCH NODES 12,0, C 1.E-20 $
  $BRANCH NODES 4,1, L 1.E-12 $
  $BRANCH NODES 1,7, L 1.E-12 $
  $BRANCH NODES 12,13, L 1.E-12 $
  $BRANCH NODES 15,16, L 1.E-12 $
  $BRANCH NODES 10,19, L 1.E-12 $
  $BRANCH NODES 11,20, L 1.E-12 $
  $BRANCH NODES 5,10, L 1.E-12 $
  $BRANCH $
  $MATRIX NODES 9,10,0, L .2165E-8, C .8670E-12, X 2.22 $
  $MATRIX NODES 11,12,0, L .2165E-8, C .8670E-12, X 2.22 $
  $MATRIX NODES 3,15,0, L .2165E-8, C .8670E-12, X 4.44 $
  $MATRIX NODES 1,0, TYPE 10 $
Y
6
.9E9 3.31E-6 -1.98E-3
1.04E9 3.13E-6 -1.55E-3
1.17E9 2.9E-6 -7.58E-4
1.3E9 2.8E-6 -1.55E-6
1.43E9 2.9E-6 7.55E-4
1.56E9 3.12E-6 1.55E-3
  $MATRIX NODES 2,0, TYPE 10 $
  $MATRIX NODES 15,0, TYPE 10 $
  $MATRIX $
  $LOADS NODES 1,0, R 50. $
  $LOADS NODES 2,0, R 50. $
  $LOADS $
```

Note: an equivalent circuit is shown in Fig. 13 with no parasitics. The parasitic values entered are negligible.

Return loss at ports 1 and 2, and insertion gain were specified at 11 frequencies between 1 GHz and 1.5 GHz. The circuit of Fig. 10 which includes the parasitic elements was described. (The data shown in Table I was fed to IMPROVE and the parasitic elements were declared variable.) Values of C_1, L_1, L_2, and L_3 were sought to meet the measured data. Sixty-seven exploratory moves and 187 pattern moves were

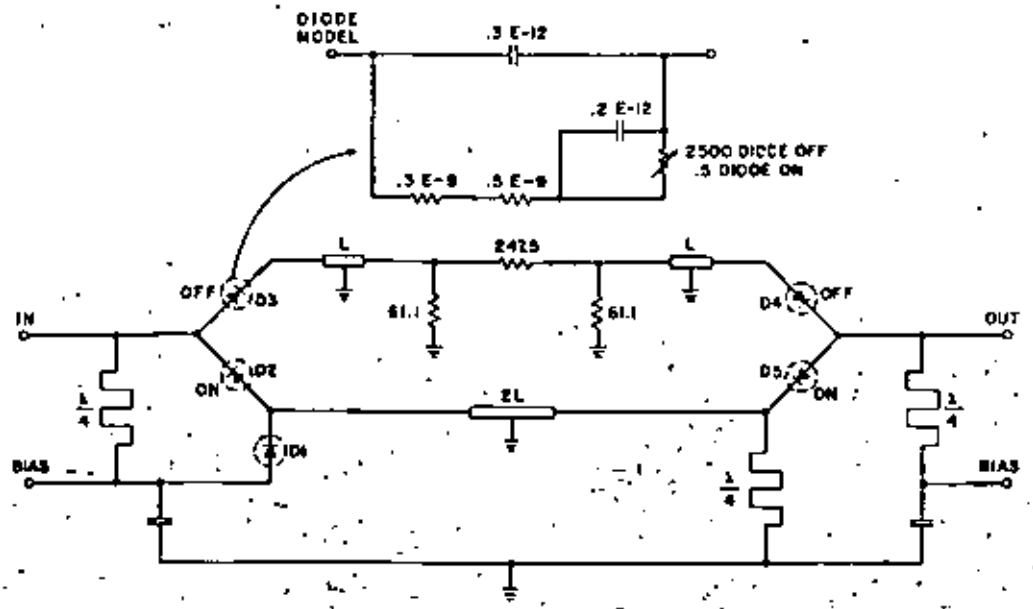


Fig. 9. 20 dB switched microwave attenuator to be analyzed by BELNAP.

TABLE II

VALUES FOR 11 FREQUENCIES OF RETURN LOSS AT PORTS 1 AND 2 AND INSERTION GAIN FROM PORT 1 TO PORT 2 OF THE CIRCUIT OF FIG. 12, MEASURED AND COMPUTED WITHOUT PARASITICS WITH BELNAP, COMPUTED WITH PARASITICS DETERMINED BY IMPROVE

Frequency ($\times 10^9$ Hz)	Desired Value (measured)	Initial Value (no parasitics)	Final Value (from IMPROVE)
Return loss at 1, dB			
1.00	29.0	34.77	29.30
1.05	27.0	31.92	27.53
1.10	25.5	29.21	25.72
1.15	23.5	27.16	23.50
1.20	22.3	24.65	22.12
1.25	21.2	23.28	21.01
1.30	20.0	22.11	20.01
1.35	19.0	21.05	19.10
1.40	18.0	20.10	18.28
1.45	17.2	19.24	17.52
1.50	16.5	18.44	16.80
Return loss at 2, dB			
1.00	20.0	23.82	19.51
1.05	17.0	20.18	16.47
1.10	12.2	15.09	11.49
1.15	13.0	9.154	13.44
1.20	26.5	17.13	29.15
1.25	31.7	26.80	32.24
1.30	27.2	45.43	24.90
1.35	23.2	33.12	21.92
1.40	21.0	27.60	20.07
1.45	19.5	24.68	18.70
1.50	18.3	22.67	17.59
Insertion Gain (1 to 2), dB			
1.00	−20.5	−20.33	−20.37
1.05	−20.5	−20.39	−20.49
1.10	−20.6	−20.74	−20.50
1.15	−23.0	−24.80	−22.86
1.20	−20.8	−21.46	−20.91
1.25	−20.7	−20.16	−20.66
1.30	−20.6	−20.61	−20.59
1.35	−20.6	−20.56	−20.58
1.40	−20.6	−20.55	−20.59
1.45	−20.6	−20.55	−20.61
1.50	−20.6	−20.56	−20.63

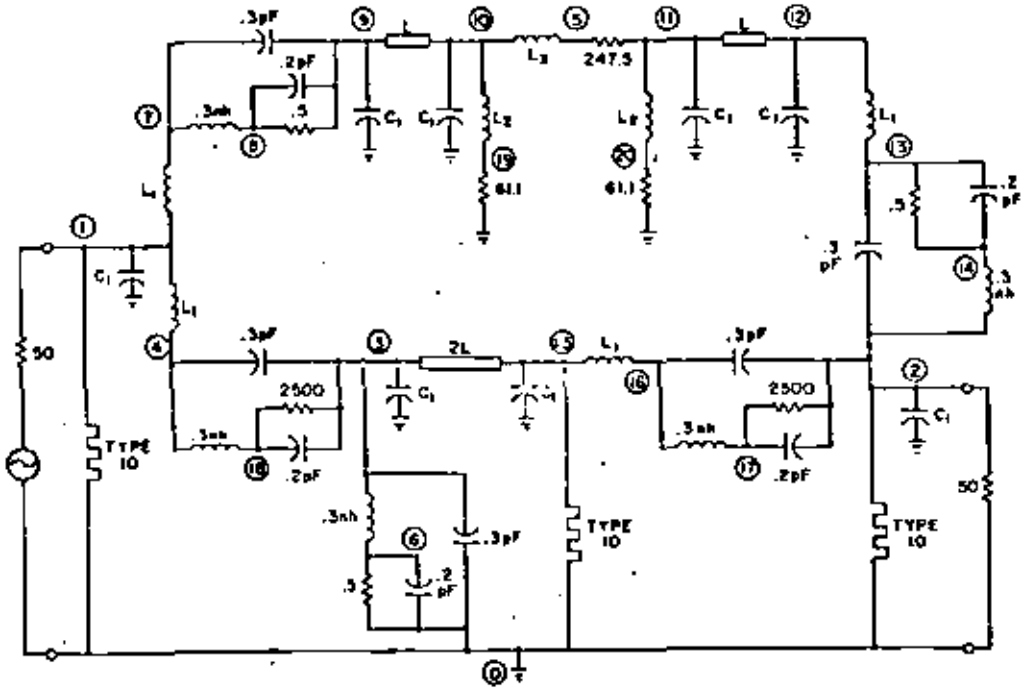


Fig. 10. Equivalent circuit of the microwave attenuator of Fig. 9 with unknown parasitics inserted.

TABLE III
VALUES OF PARASITIC ELEMENTS CALCULATED BY IMPROVE FOR THE CIRCUIT OF FIG. 13

Nodes		Element
1	0	C=1.1700E-13
2	0	C=1.1700E-13
3	0	C=1.1700E-13
15	0	C=1.1700E-13
9	0	C=1.1700E-13
10	0	C=1.1700E-13
11	0	C=1.1700E-13
12	0	C=1.1700E-13
4	1	L=3.4522E-10
1	7	L=3.4522E-10
15	16	L=3.4522E-10
10	19	L=3.1680E-10
11	20	L=3.1680E-10
5	10	L=2.3755E-09

required to converge. Seven minutes of GE635 CPU time were required. The final element values are shown in Table III. Some of the elements were constrained to be equal within the program, since they were approximately equal in the circuit layout. The response of the optimized network is compared to the response of the initial network without parasitics and the measured data. Table II shows the values of input and output return loss and insertion gain both before and after optimization. Figs. 11, 12, and 13 show the error versus frequency before and after the optimization. Good agreement is noted, indicating that the optimization has found element values that closely approximate the effects of the parasitics. A sample of printout of the analysis of the final circuit at one of the frequencies is shown in Table IV.

The above example was intended to illustrate the flexibility in modeling and topology available to the microwave engineer. A second example will be used to illustrate the design capability that such an optimization program offers. An important problem faced by microwave engineers is that of matching two arbitrary impedances. In the usual case, one impedance is the characteristic impedance of a transmission line and the second impedance is specified at a set of frequency points. This is the situation, for example, in conjugate matching of the input and output impedances of a transistor [32], [33]. One approach that has been employed with good results is the successive manual application of an analysis program to optimize the matching network [33]. The parameter values are varied and from the many responses that are recorded, the designer chooses the

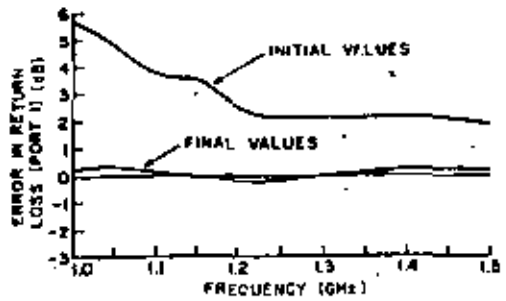
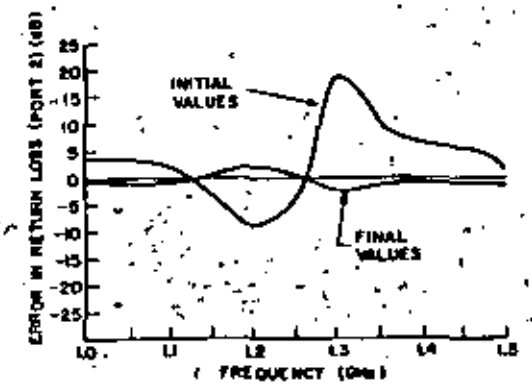


Fig. 11. Initial and final errors in return loss (port 1).



g. 12. Initial and final errors in return loss (port 2).

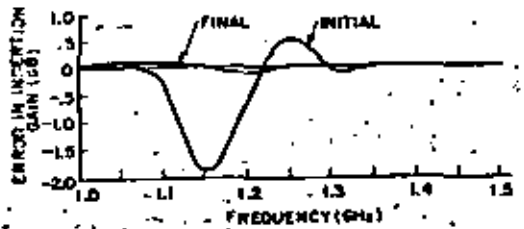


Fig. 13. Initial and final errors in insertion gain.

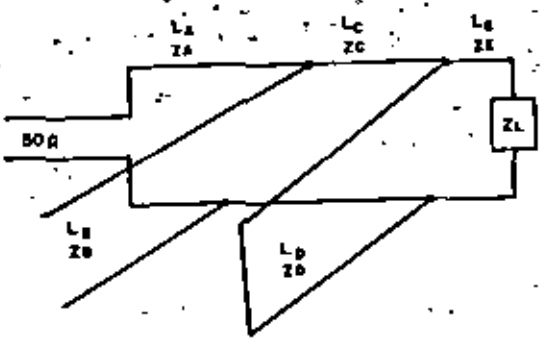


Fig. 14. Matching network.

TABLE IV

OUTPUT OF IMPROVE FOR 1.25 GHz CORRESPONDING TO FINAL CIRCUIT OF FIG. 13 WITH PARASITICS DETERMINED BY AUTOMATIC OPTIMIZATION

Frequency = 0.1250E 10

Admittance Matrix of Network

1.9874E-02	5.6214E-04
3.7010E-03	3.6714E-03
5.6214E-04	1.9680E-02
3.6714E-03	1.1105E-03

Admittance Matrix of Loads

2.0000E-02	0.
0.	0.
0.	2.0000E-02
0.	0.

Driving Point Impedance

Port	Real	Imaginary	Magnitude	Angle
1	0.4799E 02	−0.8522E 01	0.4874E 02	−0.1007E 02
2	0.4989E 02	−0.2438E 01	0.4995E 02	−0.2798E 01

Insertion Gain

Ports	dB	Angle
1 to 2	−0.2066E 02	−0.1054E 03
2 to 1	−0.2066E 02	−0.1054E 03

Voltage Ratios

	dB	Angle	Real	Imaginary
E2/E1	−0.2058E 02	−0.1003E 02	−0.1674E-01	−0.9206E-01
E1/E2	−0.2065E 02	−0.1040E 03	−0.2245E-02	−0.8999E-01

Return Loss

Port	dB
1	0.2101E 02
2	0.3224E 02

TABLE V

PARAMETER VALUES OF MATCHING NETWORK

	Initial	Final
Z_A	50	43
Z_B	50	79
Z_C	50	76
Z_D	50	74
Z_E	50	94
L_A	0.1λ	0.041λ
L_B	0.1λ	0.064λ
L_C	0.1λ	0.034λ
L_D	0.1λ	0.18λ
L_E	0.1λ	0.049λ

λ = wavelength at 1.6 GHz.

optimum. The following example will carry this procedure one step further by automatically varying the parameters until an optimum solution is achieved.

Consider the double stub matching network of Fig. 14, which will be used to match a 50 ohm line to the input impedance of a transistor specified as a set of data points. The network, as viewed from port 2 was described to the program along with the requirements. The real part of $Z_{in,1}$ was set to the real part of the transistor input impedance, and the imaginary part set to the negative of the imaginary part of the transistor impedance. The center frequency of operation was 1.6 GHz and the matching was done over a 600 MHz bandwidth. The parameters declared variable were the five line impedances and line lengths. The initial and final values of these parameters are shown in Table V. Six minutes of GE635 CPU time were required. The initial and final errors in the desired response are shown in Figs. 15 and 16.

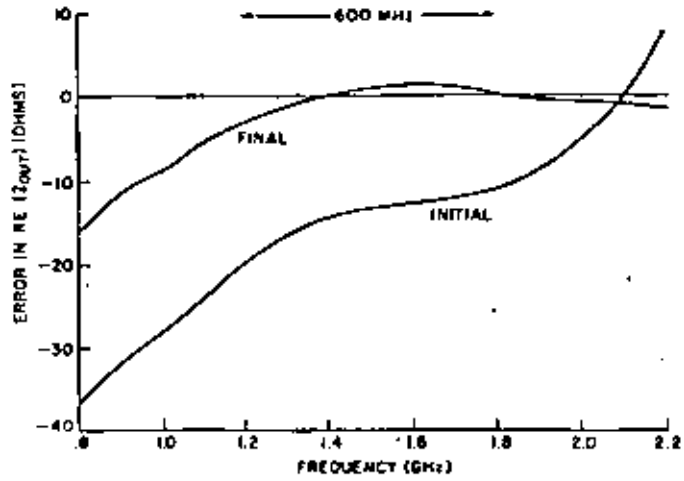


Fig. 15. Initial and final errors in Re (Z_{out}).

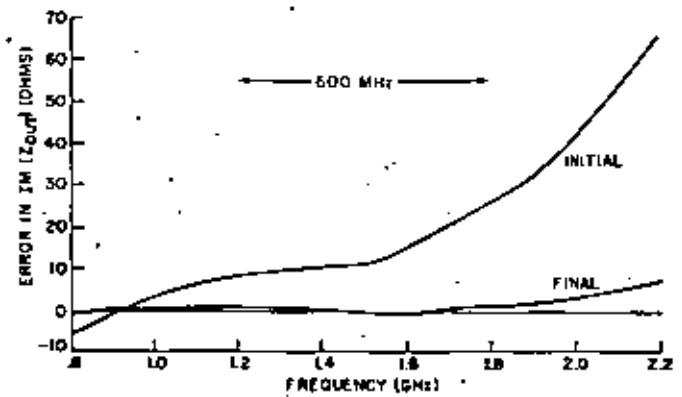


Fig. 16. Initial and final error in Im (Z_{out}).

VII. Conclusions

In this paper we have presented computer methods of analysis and design of linear networks which are suitable for microwave circuits. The methods have been implemented in two computer programs, an analysis program, BELNAP, reported on previously [16], and an analysis-optimization program, IMPROVE.

Lest the reader be misled to conclude that automatic parameter optimization is a panacea for circuit design, a few tempering comments should be made.

The success of automatic parameter optimization depends heavily upon the ability of the design engineer to avoid the pitfalls associated with it. The most serious are:

1) there is no guarantee that the method will converge to a global minimum; all it will do is go to a nearby local minimum;
2) for some functions the convergence may be painfully slow.

Because of these two stumbling blocks, a designer must use automatic optimization with great care. For example, he should come as close to a solution as possible before he applies automatic optimization. For this purpose he will use all sorts of simplifying assumptions to be able to do some analytical "ball park" calculations, use approximate synthesis methods, etc. If necessary, he will do some "by hand" searches on the computer to make sure he starts with a circuit which comes close to the specifications. Some preliminary calculations may save him considerable computer time (we mean times of the order of 30 minutes CPU on large third-generation computers).

We could give the readers the following advice.

1) Do not make too many parameters free at the same time. If you have relatively good estimates of some parameters, fix them, determine the ones you do not know, and *then* free the others. This requires more than one pass of the optimization but saves much computer time.
2) Do not use too many frequency points until you are very close to the final circuit. That is, do not use 20 frequency points when the circuit is not even in the ball park, remember every additional frequency is more computations. Start with three frequencies and increase their number when the response is close to the requirement.
3) Do not ask for unreasonable or contradictory things. Know the limitations of your circuits. Do not expect the phase to be going up when the magnitude is coming down or expect to get gain from passive elements. Do not expect to get square corners (they usually imply circuits which predict the future). Do not ask for power gains better than optimum.
4) If you are inexperienced, do not set time limits on your runs too high.

In short, do not approach automatic computer optimization naively.

In spite of the previous comments, when used properly, automatic parameter optimization is a very useful tool in the design of matching networks and directional couplers (where only approximate synthesis techniques exist). It is particularly useful in the design of nonideal networks containing active devices where parasitic and other effects have to be included and where synthesis techniques do not exist.

Considerable room for research remains in improving automatic optimization methods for microwave circuit design. Because of the pitfalls indicated above, it is very convenient to have the designer monitor the optimization as much as possible. This can be done if one uses time-shared operation or a dedicated machine. The matter of man–machine communication should receive careful consideration. In this respect the programs presented in this paper are being improved to simplify the input language which at present is somewhat complicated. It is contemplated that certain terms which network theorists prefer will be replaced by engineering quantities. (One example is the specification of lines which in microwave engineering are generally specified in terms of electrical lengths and characteristic impedance at a reference frequency, rather than R, L, G, C per unit length.) The organization of the files and programs should be specifically planned for interative operation [29]. With

respect to this item a general input-output language is being developed so that a group of programs including linear and nonlinear transient, worst case, statistical variability and tolerance programs, can access stored descriptions of circuits without having to translate the different coding schemes from program to program. Graphical aids should be provided to avoid forcing the user to go through long lists of printout.

On the numerical side, a breakthrough in efficiency is needed because most problems are too large computationally to handle on a time-shared basis. Such a breakthrough has already been made for dc circuits and single frequency ac circuits by using singular imbedding [30]. Encouraging preliminary results have been obtained for multifrequency ac and transient design. Undoubtedly the area still has room for vigorous growth. We hope this paper encourages other researchers to take the challenge.

Acknowledgment

The authors would like to acknowledge the contributions of M. P. Greer and L. A. Davieau, the writers of IMPROVE and BELNAP.

References

[1] *1620 Electronic Circuit Analysis Program (ECAP) User's Manual*, IBM Corporation, White Plains, N. Y., 1965.

[2] S. R. Sedore, "SCEPTRE: A program for automatic network analysis," *IBM Journal*, vol. 11, no. 6, pp. 627–637, November 1967.

[3] L. D. Milliman, W. A. Massena, R. H. Rickhaut, and A. C. Moog, "*CIRCUS: a digital computer program for transient analysis of electronic circuits. User's guide*," 2 vols., The Boeing Co., Washington, D. C., January 1967.

[4] L. P. McNamee and P. Potash, "A user's guide and programmer's manual for NASAP," University of California, Los Angeles, Dept. of Engineering, Rept. 68-38, August 1968.

[5] F. F. Kuo and J. F. Kaiser, Eds., *System Analysis by Digital Computer*. New York: Wiley, 1966.

[6] G. J. Herskowitz, Ed., *Computer-Aided Integrated Circuit Design*. New York: McGraw-Hill, 1968.

[7] P. E. Fleischer, "Optimization techniques in system design," in *System Analysis by Digital Computer*, F. F. Kuo and J. F. Kaiser, Eds. New York: Wiley, 1966.

[8] G. C. Temes and D. A. Calahan, "Computer-aided network optimization: the state-of-the-art," *Proc. IEEE*, vol. 55, pp. 1832–1863, November 1967.

[9] D. A. Calahan, "Computer solution of the network realization problem," *Proc. 1964 Allerton Conf. on Circuit and System Theory*, pp. 175–193.

[10] M. A. Murray-Lasso and W. D. Baker, "Computer design of multistage transistor bias circuits," *Proc. 1967 Allerton Conf. on Circuit and System Theory*, pp. 557–563.

[11] L. S. Lasdon and A. D. Warren, "Optimal design of filters with bounded, lossy elements," *IEEE Trans. Circuit Theory*, vol. CT-13, pp. 175–187, June 1966.

[12] P. O. Scheibe and E. A. Huber, "The application of Carroll's optimization technique to network synthesis," *Proc. 1965 Allerton Conf. on Circuit and System Theory*, pp. 182–191.

[13] H. A. Spang, "A review of minimization techniques for nonlinear functions," *SIAM Review*, vol. 4, pp. 343–365, 1962.

[14] D. J. Wilde, *Optimum Seeking Methods*. Englewood Cliffs, N. J.: Prentice-Hall, 1964.

[15] S. H. Brooks, "A comparison of maximum seeking methods," *J. Operations Res. Soc.*, vol. 7, pp. 430–457, 1959.

[16] M. A. Murray-Lasso, "Analysis of linear integrated circuits by digital computer using black-box techniques," in *Computer-Aided Integrated Circuit Design*, G. J. Herskowitz, Ed. New York: McGraw-Hill, 1968.

[17] R. Wyndrum, Jr., "The exact synthesis of distributed RC networks," *IEEE Conv. Rec.*, pt. 7, pp. 93–97, 1965.

[18] M. Saito, "Synthesis of transmission line networks by multivariable techniques," *Proc. Symp. on Modern Networks* (Brooklyn Polytechnic Institute), vol. 16, pp. 353–392, 1966.

[19] D. C. Youla, "Synthesis of n-ports containing lumped and distributed elements," *Proc. Symp. on Modern Networks*, vol. 16, 1966.

[20] R. W. Newcomb, *Linear Multiport Synthesis*. New York: McGraw-Hill, 1966.

[21] D. Hazony, *Elements of Network Synthesis*. New York: Reinhold, 1963.

[22] H. Ozaki and J. Ishii, "Synthesis of transmission-line networks and the design of UHF filters," *IRE Trans. Circuit Theory*, vol. CT-2, pp. 325–336, December 1955.

[23] L. Huelsman, *Circuits, Matrices and Linear Vector Spaces*. New York: McGraw-Hill, 1963.

[24] P. J. Davis, *Interpolation and Approximation*. Waltham, Mass.: Blaisdell, 1963.

[25] J. H. Ahlberg, E. N. Nilson, and J. L. Walsh, *The Theory of Splines and Their Application*. New York: Academic Press, 1967.

[26] R. Hook and T. A. Jeeves, "Direct search solution of numerical and statistical problems," *J. ACM*, vol. 8, pp. 212–229, April 1961.

[27] H. C. So, "Analysis and iterative design of networks using on-line simulation," in *System Analysis by Digital Computer*, F. F. Kuo and J. F. Kaiser, Eds. New York: Wiley, 1966.

[28] M. A. Murray-Lasso, "Black-box models for linear integrated circuits," *IEEE Trans. Education, Special Issue on Computer-Aided Design* (to be published).

[29] D. T. Ross, "The AED approach to generalized computer aided design," M.I.T. Electronic Syst. Lab., Cambridge, Mass., Rept. ESL-R-305, April 1967.

[30] B. B. Kozemchak and M. A. Murray-Lasso, "Computer-aided circuit design by singular imbedding," *Bell Sys. Tech. J.*, vol. 48, pp. 275–315, January 1969.

[31] R. M. Fano, "Theoretical limitations on broadband matching of arbitrary impedances," *J. Franklin Institute*, vol. 249, pp. 57–83, 139–155.

[32] P. E. Emery, M. O'Hagen, and S. D. Nolte, "Optimal design of matching networks for microwave transistor amplifiers," *1966 G-MTT Internatl. Symp. Digest*, pp. 101–106.

[33] V. G. Gelnovatch and T. F. Burke, "Computer-aided design of wide-band integrated microwave transistor amplifiers on high dielectric substrates," *IEEE Trans. Microwave Theory and Techniques*, vol. MTT-16, pp. 429–439, July 1968.

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

OPTIMACION DE SISTEMAS FISICOS

OPTIMACION DE MECANISMOS

DR. JORGE ANGELES ALVAREZ

MARZO, 1981

"SINTESIS DE MECANISMOS GENERADORES DE FUNCION CON TRANSMISION OPTIMA"

J. Angeles
División de Estudios de Posgrado
de la Facultad de Ingeniería, UNAM.
Apdo. Postal 70-256, México 20, D.F.

J.L. Hernández
Ford Motor Company, S.A.
Cuatitlán, Edo. de México.

Abstract

The linkage-synthesis problem for function generation is approached from the viewpoint of system optimization. An objective function is proposed, whose minimization guarantees an optimal transmission. The optimization method resorted to is the Complex Method, which showed excellent results as to speed of convergence. The procedure is illustrated with an example of application to a planar linkage, but the extension to space linkages is straightforward.

Resumen

El problema de la síntesis de mecanismos generadores de función es abordado desde el punto de vista de la optimación de sistemas. Se propone una función objetivo cuya minimización garantiza una transmisión óptima. El método de optimación utilizado es el Cómplex, que mostró excelentes resultados en cuanto a rapidez de convergencia. Se ilustra este procedimiento con un ejemplo de aplicación a un mecanismo plano; pero la extensión a mecanismos espaciales es inmediata.

Introducción

En el problema de síntesis de mecanismos para generación de funcion se trata de hallar un conjunto de valores $p_1,\ldots,p_n$ de parámetros de un mecanismo de topología dada, que produzcan un conjunto de pares de valores entrada-salida $\{(\psi_i,\phi_i)\}_1^n$, donde ψ y ϕ son variables que representan la entrada y la salida del mecanismo, respectivamente. Estas variables pueden ser desplazamientos angulares o lineales. Igualmente, los parámetros p_i $(i=1,\ldots,n)$ pueden ser distancias o ángulos. Por ejemplo, dado el mecanismo de la Fig 1, los parámetros son las longitudes a_1, a_2, a_3 y a_4 de los eslabones, mientras que la entrada y la salida son los ángulos ψ y ϕ, respectivamente. La ecuación que relaciona los parámetros con las variables de entrada y de salida es la ampliamente conocida ecuación de Frendenstein (ref. 1):

$$k_1-k_2\cos\phi+k_3\cos\psi+\cos(\psi-\phi)=0 \qquad (1)$$

donde

$$k_1=\frac{a_3^2-a_1^2-a_2^2-a_4^2}{2a_2a_4},\quad k_2=\frac{a_1}{a_2},\quad k_3=\frac{a_1}{a_4} \qquad (2)$$

Dado que la ec (1) contiene tres parámetros independientes, k_1, k_2 y k_3, sólo se puede generar 3 pares de valores (ψ_i,ϕ_i) por medio de ella. En efecto, sustituyendo esos 3 pares de valores en la ec. (1) se obtiene el siguiente sistema lineal de ecuaciones algebraicas.

$$\underline{A}\underline{k}=\underline{b} \qquad (3)$$

donde

$$\underline{A}=\begin{bmatrix}1 & -\cos\phi_1 & \cos\psi_1\\ 1 & -\cos\phi_2 & \cos\psi_2\\ 1 & -\cos\phi_3 & \cos\psi_3\end{bmatrix},\quad \underline{k}=\begin{bmatrix}k_1\\ k_2\\ k_3\end{bmatrix},\quad \underline{b}=\begin{bmatrix}-\cos(\psi_1-\phi_1)\\ -\cos(\psi_2-\phi_2)\\ -\cos(\psi_3-\phi_3)\end{bmatrix} \qquad (4)$$

El sistema de ecuaciones (3) se puede resolver por varios métodos, de los cuales el más eficiente es el de Gauss o descomposición LU (ref. 2). En caso de necesitar generar más de 3 pares de valores (ψ_i,ϕ_i), por ejemplo m>3, la matriz $\underline{A}$ de la ec. (3) resulta rectangular, de mx3 y el sistema de ecuaciones es sobredeterminado. En este caso, generalmente no existe una solución que satisfaga simultáneamente todas las ecuaciones, pero se puede hallar aquel valor de $\underline{k}$, $\underline{k}_0$, que minimice la norma cuadrática del error

$$\underline{e}=\underline{A}\underline{k}-\underline{b} \qquad (5)$$

La solución $\underline{k}_0$ se puede hallar de una manera muy eficiente mediante el método de refleciones de Householder (ref.3), que ya ha sido aplicado con éxito a la síntesis de mecanismos para un número excesivo de puntos de precisión (ref.4).

Volviendo a la ec. (3), y considerando el caso en el que se tenga igual número de puntos de precisión que de parámetros a determinar, esto es, 3, es bien sabido (ref.5) que el sistema de ecuaciones tiene una solución única, por lo que el mecanismo sintetizado será único también. Por esta razón, es posible que este mecanismo tenga una operación pobre, como por ejemplo, una mala transmisión. La transmisión de un mecanismo es una cualidad asociada a la ventaja mecánica del mismo, o sea, la relación entre el momento obtenido a la salida y el momento

sumunistrado a la entrada. Una forma de cuantificar esa transmisión, ampliamente aceptada, es mediante el llamado ángulo de transmisión, μ, que aparece en la Fig 1. Mientras este ángulo más próximo esté de un valor de 90° o de 270°, mayor será la ventaja mecánica, anulándose ésta cuando aquel ángulo vale 0° o 180°. Es deseable, entonces, que el ángulo de transmisión adquiera valores que se desvíen lo menos posible de 90° o de 270°, según el caso. En seguida se discute como modificar la ecuación de Freudenstein para poder convertir el problema en uno de optimación.

El problema de síntesis óptima

La ecuación de Frendenstein, tal como aparece en la ec. (3), no permite ninguna optimación. Sin embargo, si los ángulos de entrada y de salida se miden no desde la línea determinada por el eslabón 1, sino desde líneas que forman ángulos α y β con esta línea, respectivamente, la ecuación de Freudenstein se transforma en

$$k_1-k_2\cos(\phi+\beta)+k_3\cos(\psi+\alpha)+\cos(\phi-\psi-\alpha+\beta)=0 \qquad (6)$$

donde, debido a que los ángulos α y β están aún indeterminados, se tiene un conjunto de 5 incógnitas. Para el problema de generación de 3 pares de valores entrada-salida, (ψ_i,ϕ_i), entonces, se puede asignar libremente valores a dos de esas incógnitas. Si esas dos incógnitas son α y β, éstas se pueden escoger de manera que produzcan el mejor ángulo de transmisión, μ. Una función positiva definida cuya minimización produce valores de μ próximos a 90° o a 180° es

$$z = \frac{1}{2\pi}\int_0^{2\pi}\cos^2\mu\, d\phi \qquad (7)$$

donde se ha supuesto que el eslabón de entrada, 2, gira vuelta completa. La condición que deben satisfacer las longitudes de los eslabones, para que el de entrada gire vuelta completa, está dada por las desigualdades (ref. 1,p.63)

$$a_2>a_1 \qquad (8)$$

$$a_4>a_1 \qquad (9)$$

$$a_3+a_4>a_2-a_1 \qquad (10)$$

$$a_3+a_4>a_1+a_2 \qquad (11)$$

El ángulo de transmisión está dado por (ref.6):

$$\cos\mu=\frac{a_2^2-a_3^2-a_4^2+a_1^2-2a_1a_2\cos(\phi+\alpha)}{2a_3a_4} \qquad (12)$$

Así, el problema de optimación resultante es el siguiente: "Minimizar z dada por la ec.(7) sujeta a las restricciones de igualdad (3) y a las de desigualdad (8) a (11)".

Los métodos disponibles para resolver el problema de optimación propuesto son básicamente de dos tipos:

i) métodos de funciones de penalización y ii) métodos directos. Los métodos de funciones de penalización consisten en transformar el problema dado, que contiene restricciones de desigualdad, en una secuencia de problemas sin este tipo de restricciones, y hallar los valores óptimos de las variables de decisión (α y β en este caso) para cada uno de esos problemas. El óptimo del problema original, que contiene restricciones de desigualdad, se obtiene por extrapolación (ref.7). El método de funciones de penalización ya se ha usado con éxito en la síntesis de mecanismos (ref.8). Los métodos directos manejan las restricciones de desigualdad directamente, esto es, no transforman el problema en uno sin este tipo de restricciones.

En cualquier caso el método de optimación a seguir dependerá de cuántas derivadas, con respecto a las variables de decisión, se tengan disponibles, esto es

i) ninguna derivada se puede calcular,

ii) se pueden calcular sólo primeras derivadas

iii) se tiene acceso a derivadas hasta de orden 2.

Dentro de los métodos aplicables al primer caso se tiene,el de Powell y el Cómplex. El de Powell (ref.9) calcula el óptimo de una función sin restricciones de disiqualdad y sin requerir de derivadas, mientras que el Cómplex (ref.10) calcula el óptimo de una función sujeto a restricciones de desigualdad, también sin requerir de derivadas. Ambos métodos ya han sido probados en la síntesis de mecanismos (refs.11 y 12).

Si se tiene acceso a primeras derivadas, se tienen los métodos de gradiente y de cuasi Newton (refs.13 y 14). Finalmente, si se dispone hasta de segundas derivadas, se puede aplicar el método de Newton-Raphson (ref.15) funciones de penalización para calcular las raíces del gradiente de la función objetivo a la que se ha aumentado las funciones de penalización adecuadas.

En el ejemplo que sigue se utilizó el método Cómplex.

Ejemplo de aplicación

Se desea sintatizar un mecanismo plano RRRR como el que aparece en la Fig 1, que tenga una transmisión óptima, en el sentido de que minimice el valor RMS de cosμ, dado

por la ec. (7), de manera que su eslabón de entrada gire vuelta completa -desigualdades (8) a (11)-y genere la función

$$\psi_1 = 30°, \quad \phi_1 = 45°$$
$$\psi_2 = 150°, \quad \phi_2 = 60°$$
$$\psi_3 = 270°, \quad \phi_3 = 90°$$

Como la ecuación de Freudenstein sólo contiene 3 parámetros independientes, a una de las 4 longitudes a_i se le puede dar un valor arbitrario. Hágase, por ejemplo

$$a_1 = 1$$

ya que cualquiera que sea la solución del problema, esta longitud siempre es positiva. La ecuación de Freudenstein se utiliza, desde luego, en la forma de la ec.(6), con α y β como variables de decisión, que permiten la optimación del mecanismo.

Con el objeto de simplificar los cálculos, escríbanse la función objetivo z y las restricciones (8) a (11) en términos de los parámetros k_i. Así,

$$a_2 = \frac{1}{k_2}$$
$$a_3 = \frac{\sqrt{D}}{k_2 k_3}$$
$$a_4 = \frac{1}{k_3} \qquad (13)$$
$$z = \frac{E}{2D}$$

donde

$$D = 2k_1k_2k_3 + k_3^2 + k_2^2 + k_2^2k_3^2 \qquad (14)$$

y

$$E = 2(k_1k_3 + k_2)^2 + k_3^4 \qquad (15)$$

La matriz $\underline{A}$ y el vector $\underline{b}$ se transforman claramente en

$$\underline{A} = \begin{bmatrix} 1 & -\cos(\phi_1+\beta) & \cos(\phi_1+\alpha) \\ 1 & -\cos(\phi_2+\beta) & \cos(\phi_2+\alpha) \\ 1 & -\cos(\phi_3+\beta) & \cos(\phi_3+\alpha) \end{bmatrix}, \quad \underline{b} = \begin{bmatrix} -\cos(\phi_1-\phi_1-\alpha+\beta) \\ -\cos(\phi_2-\phi_2-\alpha+\beta) \\ -\cos(\phi_3-\phi_2-\alpha+\beta) \end{bmatrix} \qquad (16)$$

Las restricciones (8) a (11) se reducen a

$$k_2 < 1$$
$$k_3 < 1$$
$$k_4[k_2 - k_3 + D^{1/2} - k_2k_3] > 0 \qquad (17)$$

Adicionalmente, limítense los valores de α y β entre 0 y 2π. Así se tienen además las siguientes desigualdades

$$0 \leq \alpha \leq 2\pi$$
$$0 \leq \beta \leq 2\pi \qquad (18)$$

Para la solución del problema de optimación propuesto se utilizó el paquete OPTIM (ref.16), que dio como solución los siguientes valores numéricos

$a_2 = 0.3774$ u. de longitud
$a_3 = 0.7450$ u. de longitud
$a_4 = 1.001$ u. de longitud
$\alpha = 227.6°$
$\beta = 157.1°$

El mecanismo correspondiente se muestra en la Fig 2 y la curva μ vs. ϕ, en la Fig 3.

Conclusiones y recomendaciones

El método Cómplex mostró una buena rapidez de convergencia, pues el número máximo de iteraciones requerido fue de 23. Sin embargo, debe tenerse en cuenta que el problema presentado sólo contiene dos variables de decisión. Si el número de estas variables aumenta es posible que el método presente dificultades para converger. En ese caso, debe ensayarse el método de Newton-Raphson con funciones de penalización y amortiguamiento, lo cual acelera notablemente la convergencia. La función objetivo utilizada es cuadrática y positiva definida, lo cual hace que tenga primeras y segundas derivadas continuas, por lo que se facilita su uso para el método de Newton-Raphson. El método Cómplex no requiere tal continuidad en esas derivadas y se pudo haber usado, en cambio, otra función objetivo como

$$z = \max|\cos\mu|$$

El hecho de haber usado una función cuadrática fue motivado por razones de comparación, en caso de que se desee ensayar con el método de Newton-Raphson, por ejemplo.

Referencias

1. Angeles J., Análisis y Síntesis Cinemáticos de Sistemas Mecánicos, Ed. Limusa, S.A., México, D.F., 1978, p. 45.

2. Forsythe G. y Moler C.B., Computer Solution of Linear Algebraic Systems, Prentice Hall, Inc., Englewood Cliffs, N.J., 1967, p. 27

3. Moler C.B., Matrix Eigenvalue and Least Square Computations, Computer Science Department, Stanford University, Stanford Cal.,1973, pp. 4.1-4.15

4. Angeles J., "Optimal Synthesis of linkages using Householder reflections", Proceedings of the Fifth World Congress on the Theory of Machines and Machanisms, vol. 1, Montreal, Canadá, julio 8-13, 1979, pp. 111-114.

5. Finkbeiner D.T., Introduction to Matrices and Linear Transformations, W.H. Freeman and Co., San Francisco, 1966, p. 99

6. Denavit J. y R.S. Hartenberg, Kinematic Synthesis of Linkages, McGraw-Hill Book Co., N. York, 1964, p. 319.

7. Aoki M., Introduction to Optimization Techniques. Fundamental and Applications of Nonlinear Programming, The MacMillan Co., N. York, 1971, pp. 199-204

8. Alizade R.I., Novruzbekov I. G. y Sandor G.N., "Optimization of Four-bar function generating mechanisms using penalty functions with inequality and equality constraints", Mechanism and Machine Theory, vol. 10, 1975, pp. 327-336

9. Powell M. J. D., "An efficient method for finding the minimum of a function of several variables without calculating derivatives", Computer Journal, vol. 7, No. 4, 1964, pp. 155-162.

10. Box M.J., "A new method of constrained optimization and a comparisson with other methods", Computer Journal, vol. 8, 1965, pp. 42-52.

11. Suh C.H. y C.W. Radcliffe, Kinematics and Mechanisms Design, John Wiley and Sons, N. York, 1978, pp. 215-217.

12. Dukkipati R.V., Sankar S. y Osman M.O.M., "On the use of complex method of constrained optimization in linkage synthesis", Proceedings of the Fifth World Congress on the Theory of Machines and Mechanisms, vol. 1, julio 8-13, 1979, Montreal, Canadá, pp. 382-387.

13. Zoutendijk G., Methods of Feasible Directions, Elsevier Publishing Co., Amsterdam, 1960.

14. Fox R.L. and Gupta K. C., "Optimization technology as applied to mechanism design" J. Eng. Ind., Trans. ASME, Serie B, vol. 95, mayo 1973, pp. 657-663.

15. Isaacson E. y Keller H. B., Analysis of Numerical Methods, John Wiley and Sons, Inc., N. York, 1966, pp. 115-119

16. Evans L. B., Optimization Techniques for Use in Analysis of Chemical Processes, A Set of Notes, Massachussetts Institute of Technology, Cambridge (USA), 1971.

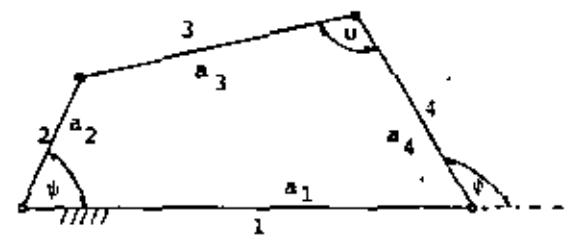


Fig 1. Mecanismo plano RRRR

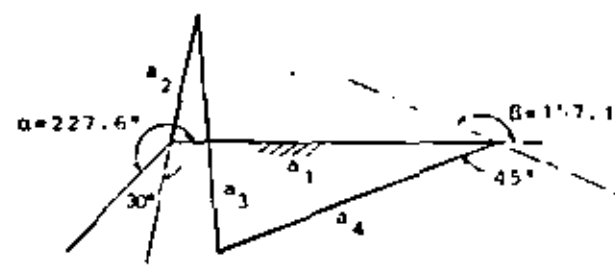


Fig 2. Mecanismo RRRR plano generador de función con transmisión óptima.

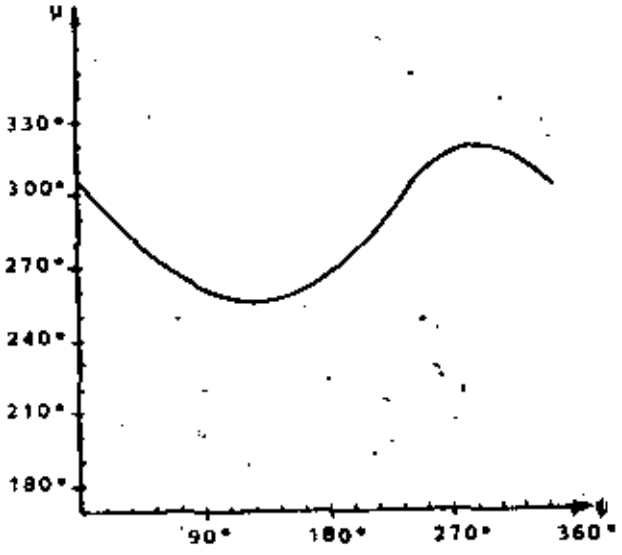


Fig 3. Curva de μ vs. ψ

Proceedings of the Fifth World Congress on Theory of Machines and Mechanisms — 1979
Published by The American Society of Mechanical Engineers

OPTIMAL SYNTHESIS OF LINKAGES USING HOUSEHOLDER REFLECTIONS

J. Angeles, Professor
National University of Mexico (UNAM)
Mexico, D. F., Mexico

ABSTRACT

The unconstrained overdetermined problem of kinematic linkage synthesis is solved in an efficient way using Householder reflections. The problem formulation leads to a system of either linear or nonlinear equations in more equations than unknowns. The linear problem is solved directly by application of a finite number of successive reflections to the space of unknowns with the purpose of taking the system of equations into upper triangular form, which allows for the computation of the unknowns by back substitution. The nonlinear problem is solved via the Newton-Raphson method which computes, at each iteration, the correction to the vector of unknowns as the least-square solution to an overdetermined linear system in exactly the same way as described before for linear problems. Introduction of the said method produces accurate results in relatively short processing times, as shown in the examples presented.

ZUSAMMENFASSUNG

Das uneingeschränkte und überbestimmte Problem der kinematischen Getriebesynthese wird effizient gelöst mit Hilfe der Householder-Spiegelungen. Die Problemstellung leitet zu einem System von entweder linearen oder nichtlinearen Gleichungen mit mehr Gleichungen als Unbekannten. Das lineare Problem wird direkt gelöst mittels Anwendung einer finiten Zahl aufeinanderfolgenden Spiegelungen zum Raum der Unbekannten mit dem Ziel des Überführens des Gleichungssystems zu einer höheren dreieckigen Form, welche die Rechnung der Unbekannten durch Rücksetzung erlaubt. Das nicht-lineare Problem wird mittels der Newton-Raphson-Methode gelöst, die zu jeder Iteration die Besserungen der Unbekannten aus der wenigsten Quadraten-Lösung zu einem überbestimmten linearen Gleichungssystem errechnet, auf der gleichen Weise wie bei der Methode für lineare Systeme schon beschrieben wurde. Die Einführung dieser Methode führt zu deutlichen Erfolgen in relativ kurzen Prozessierzeiten, wie mittels der eingeschlossenen Beispielen gezeigt wird.

NOMENCLATURE

$\underline{\underline{A}}$: upper-case underlined character, an mxn matrix.
$\underline{\underline{A}}^{-1}$: the inverse of $\underline{\underline{A}}$, when $\underline{\underline{A}}$ is square and nonsingular
$\underline{\underline{A}}^{T}$: the transpose of $\underline{\underline{A}}$
$\underline{a}$: lower-case underlined latin character, an m-dimensional vector
$|a|$: the absolute value of a, when a is real; the modulus of a, when a is complex.
$||\underline{a}||$: the Euclidean norm of vector $\underline{a}$, i.e. the square root of the sum of the squares of its components
$\det\underline{\underline{A}}$: the determinant of the square matrix $\underline{\underline{A}}$
$\underline{f}(\underline{x})$: an m-dimensional vector function of the n-dimensional vector argument $\underline{x}$
$\underline{\underline{f}}'(\underline{x})$: the Jacobian mxn matrix of $\underline{f}$ with respect to $\underline{x}$

PROBLEM FORMULATION

The equations arising in the realm of kinematic synthesis of linkages constitute either linear or nonlinear algebraic[1] systems (1,2), whose unknowns are the geometric parameters (lengths and angles) of the linkage. If these parameters are arranged within the n-dimensional vector $\underline{x}$, the said equations are of the form

$$\underline{\underline{A}}\,\underline{x}=\underline{b} \qquad (1)$$

where $\underline{\underline{A}}$ and $\underline{b}$ are a known mxn matrix and an m-dimensional known vector, respectively, when the system is linear. If it is nonlinear, then the synthesis equations are of the form $\underline{f}(\underline{x})=\underline{0}$ (2)

$\underline{f}$ being an m-dimensional vector containing a set of m scalar functions $f_i(\underline{x})$ whose arguments are the unknown parameters of the linkage. When the number of specified conditions to be met by the linkage matches that of the unknowns, matrix $\underline{\underline{A}}$ in (1) is square and vector $\underline{f}$ is of dimension n. In most technical problems, however, the number of prescribed conditions surpasses that of geometric parameters available, the linkage synthesis problem thus leading to an overdetermined system of equations. This class of systems in general does not admit an exact solution, but it is possible to find a vector $\underline{x}$ that renders the quadratic error a minimum. Thus, the least-squares problem can be stated as:

"Find the value of x that minimizes the Euclidean norm[2] of either $\underline{A}\underline{x}-\underline{b}$, or that of $\underline{f}(\underline{x})$, depending on whether the system is linear or nonlinear".

The linear overdetermined system (1) admits a unique solution $\underline{x}$ that renders $||\underline{\underline{A}}\underline{x}-\underline{b}||$ a minimum, provided $\underline{\underline{A}}$ is of full rank, i.e. if $\text{rank}\,\underline{\underline{A}}=n$. This value is given as (3)

$$\underline{x}=(\underline{\underline{A}}^{T}\underline{\underline{A}})^{-1}\underline{\underline{A}}^{T}\underline{b} \qquad (3)$$

where $(\underline{\underline{A}}^{T}\underline{\underline{A}})^{-1}\underline{\underline{A}}^{T}$ is called a "Moore-Penrose generalized

[1] Algebraic as opposed to differential or integral equations.

[2] See the nomenclature for the definition of this term.

inverse of A^T. An extensive treatment of the linear least-squres problem is found in (4).

The nonlinear problem may admit multiple local minima; these can be found by application of the Newton-Raphson method (5), which at each iteration, computes the correction vector Δx_k as the least-square solution to the overdetermined linear system

$$f'(x_k)\Delta x_k = -f(x_k) \qquad (4)$$

This is a system like that appearing in eq.(1). Thus, its least-square solution is

$$\Delta x_k = -[f'(x_k)^T f'(x_k)]^{-1} f'(x_k)^T f(x_k) \qquad (5)$$

The new value of the unknown vector is then

$$x_{k+1} = x_k + \Delta x_k \qquad (6)$$

The procedure is stopped when the Euclidean norm of the correction vector is sufficiently small within the imposed accuracy, i.e. when

$$||\Delta x|| \leq \varepsilon \qquad (6)$$

ε being a "small" real positive number. The problem thus, whether linear or nonlinear, reduces to compute the minimizing value x_0 given by eq. (3). An efficient way of computing this value, outlined next, does not require to invert any matrix. The computation is done by application of Householder reflections.

HOUSEHOLDER REFLECTIONS

An extensive account of this topic can be found in the specialized literature (4,6). For this reason, this theory is not treated here. A Householder reflection is a linear, improper orthogonal and symmetric transformation, i.e., if H is its mxm matrix representation, then

$$H = H^T = H^{-1}, \det H = -1 \qquad (8)$$

When n such tranformations are defined suitably, their effect on matrix A appearing in eq. (1) is to take it into upper triangular form. This way, the transformed equations are equivalent to the following

$$Ux = c \qquad (9)$$

$$Ox = d \qquad (10)$$

where U is an upper triangular nxn matrix and O is the (m-n)xn zero matrix, c and d being n-and (m-n)-dimensional vectors, with $d \neq 0$. Thus, eq. (9) is determined and can readily be solved by back substitution, its solution x_0 being the least-square solution to the overdetermined system. Eq. (10) is inconsistent and $||d||$ represents the Euclidean norm of the error in the approximation. Since the original system (1) is transformed into (9),(10) via a succession of orthogonal transformations, the error in the transformed coordinates, d, has the same Euclidean norm as that in the original coordinates. Hence, $||d||$ is the error associated with the original system.

APPLICATIONS TO KINEMATIC LINKAGE SYNTHESIS

Although in many practical applications the problems of linkage synthesis involve inequality constraints, still a considerably large class of synthesis problems are unconstrained. Moreover, efficient optimization techniques exist that handle inequality constraints by introducing suitable penalty functions (7), thus turning the problem an unconstrained one. For these reasons, the study of unconstrained optimization problems is of substantial technical interest. Applications to linkage synthesis problems are next illustrated with two examples.

Example 1. Synthesis of an RSSR function generator

The layout of an RSSR linkage, shown in Fig 1, indicates the different geometric parameters of this linkage; a_1, a_2, a_3, are the lengths of the output-, coupler-and input links, respectively; a_4 is the distance between the axes of the input and the output links; α_4 is the angle between the aforementioned axes, positve about ED; s_1 and s_4 are distances of points C and O along the axes of the output-and the input links, respectively. All over, the sign convention of Denavit and Hartenberg (1,pp.344-345) is observed. The input angle is ψ and the output angle is ϕ. For matching six pairs of input-output values (ψ_j,ϕ_j) with this linkage, Denavit and Hartenberg (1,pp.358-362) established the following relation

$$k_1\cos\phi_j + k_2\sin\phi_j - k_3\cos\psi_j + k_4\sin\phi_j + k_5(\sin\phi_j\cos\psi_j - \cos\alpha_4\sin\psi_j\cos\phi_j) + k_6 = \cos\phi_j\cos\psi_j + \cos\alpha_4\sin\phi_j\sin\psi_j \qquad (11)$$

where

$$k_1 = \frac{a_4 + s_4\sin\alpha_4\tan\phi_0}{a_3}, \quad k_2 = \frac{s_4\sin\alpha_4 - s_4\tan\phi_0}{a_3}$$

$$k_3 = \frac{a_4}{a_1\cos\phi_0}, \quad k_4 = \frac{s_1\sin\alpha_4}{a_1\cos\phi_0}, \quad k_5 = \tan\phi_0 \qquad (12)$$

$$k_6 = \frac{a_1^2 - a_2^2 + a_3^2 + a_4^2 + s_1^2 + s_4^2 + 2s_1 s_4\cos\alpha_4}{2a_1 a_3\cos\phi_0}$$

α_4 being assigned.

In the latter definitions, ϕ_0 measures the location of the zero of the output dial from the dotted line passing through C, parallel to line ED, as shown in Fig 1.

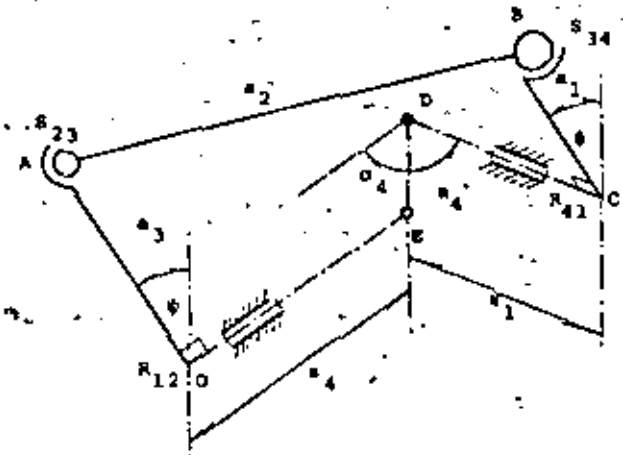


Fig 1 An RSSR linkage

For six precision-point synthesis, eqs. (11) yield a system of six linear equations in six unknowns which, when nonsingular, produces unique values $k_1, k_2, \ldots, k_6$. With these values known, the linkage parameters are computed from eqs. (12) for a given value of a_1. If more than six precision points are required, however, the system becomes overdetermined,

in which case an efficient method to obtain its least-square solution is via Householder reflections.

In (8), Suh and Mecklenburg solve the over-determined unconstrained problem of this linkage with 19 prescribed input-output values. For comparison purposes, the solution developed in this example makes use of the same prescribed values. These are shown in Table 1

The method employed in (8) is that of Powell's (9), which does not require the computation of derivatives and leads quickly to convergence for quadratic functions of the independent variables. At this point, two remarks are in order: First, the derivatives of the synthesis equations are easily computed from either Denavit and Hartenberg's formulation, eqs. (11), or from Suh and Radcliffe's formulation (2), the first one being advantageous because of producing a linear system of equations. Second: The objective function of Suh and Mecklenburg's (8) is quadratic in the synthesis which, in turn, are quadratic in the independent variables; thus, their objective function is quartic in the independent variables, for which reason the quick convergence properties of Powell's method are not fully utilized. Furthermore, squaring the synthesis functions may introduce spurious local minima, as is apparent form the fact that three optimal solutions are reported in (8).

TABLE 1. Specified input-output pairs for the synthesis of the RSSR function generating linkage.

	ϕ(degrees)	ϕ(degrees)
1	0.0	0.0
2	5.0	2.4
3	10.0	5.1
4	15.0	8.2
5	20.0	11.5
6	25.0	15.2
7	30.0	19.1
8	35.0	23.3
9	40.0	27.7
10	45.0	32.3
11	50.0	37.2
12	55.0	42.3
13	60.0	47.5
14	65.0	53.0
15	70.0	58.7
16	75.0	64.6
17	80.0	70.9
18	85.0	78.0
19	90.0	90.0

One advantage of using Householder reflections is that no explicit squaring is required, and the unique solution is obtained directly by the application of n(=6) reflections. Another advantage is that, since less computations are required, as compared to Powell's method, the round-off error is lowered. The approximation error obtained using each method is shown in Table 2.

The root mean square errors were essentially the same: that obtained by Powell's method was 0.00185269; whereas the one obtained by Householder reflections, 0.00182254. However, the differences in the resulting linkage parameters were more notorious. These are

Solution by Powell's method	Solution by Householder's method
s_1 = 1.253803	s_1 = 0.911269
s_2^1 = 2.759566	s_2^1 = 2.620568
s_3^1 = 0.495003	s_3^1 = 0.803577
s_4 = 2.262110	s_4 =-1.186240
s_4^1 =-1.375270	s_4^1 =-2.417556

In this problem, a_4 was set equal to 1, whereas α_4, equal to 90°.

TABLE 2. Approximation error in overdetermined RSSR linkage synthesis

	APROXIMATION ERROR USING POWELL'S METHOD (degrees)	APROXIMATION ERROR USING HOUSEHOLDER'S METHOD (degrees)
1	0.00000000	0.01420369
2	-.00120000	0.00100785
3	-.03060000	-.03515343
4	0.02330000	0.01590827
5	-.02680000	-.03451603
6	0.03760000	0.02596072
7	0.01600000	0.01079286
8	0.03830000	0.03440434
9	0.01520000	0.01196148
10	-.03790000	-.04106664
11	-.00640000	-.00968497
12	0.02270000	0.01930349
13	-.04220000	-.04497542
14	-.00020000	-.00155544
15	0.03350000	0.03434700
16	0.01600000	0.01324614
17	0.00020000	0.00139267
18	-.01250000	-.01970407
19	0.02980000	0.00412793

Example 2. Synthesis of the RR plane dyad for rigid-body guidance

A rigid body (shaded rectangle) appears in Fig 2, in "reference" configuration C_0 and in a different configuration C_j. Each configuration is defined by the position of a point, R, and angle, θ. In that figure, O represents the origin of the complex plane, and the arrows represent complex numbers associated with the location of the labelled points. The purpose of this class of synthesis problem is to locate point A whose reference and successive positions, A_0, A_j (j=1, ...,n) lie on a circumference centered at B, for which reason, A_0 and B are called, respectively, "circular" and "central" points, within the Burmester Theory (10). Thus, AB_0 can constitute a rigid link to guide the rigid body. This is an RR plane dyad.

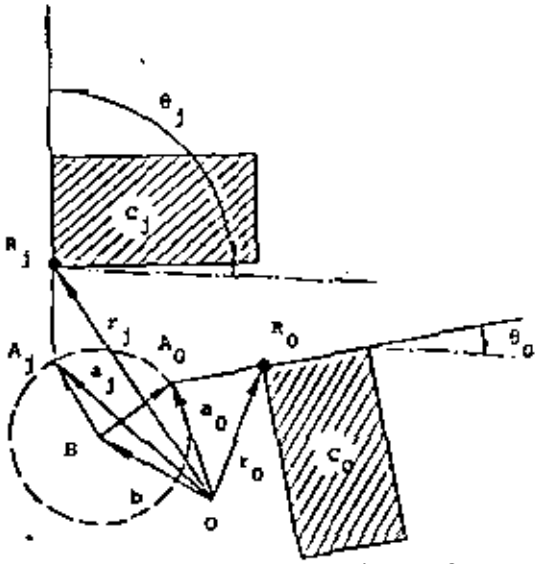


Fig 2 An RR plane dyad to guide a rigid body through n successive configurations

The constancy of the length of line BA throughout its n configurations leads to

$$|e^{i\theta'_j}(a_0-r_0)+r_j-b|^2=|a_0-b|^2, j=1,\ldots,n \qquad (13)$$

where $\theta'_j \equiv \theta_j-\theta_0$. Eqs. (13) constitute the synthesis equations for this problem, a_0 and b being the unknowns. It is well known (2,p.146) that this problem allows to conduct a rigid body through five specified configurations. Some technical problems, however, may require to guide the body through more than five configurations, as shown in Table 3. Different syntheses were obtained for these, starting from the first 6 configurations, then adding the next ones,one at each time, until the 16 configurations were included

TABLE 3. Successive configurations of a rigid body

j	x_j(cm)	y_j(cm)	θ_j(degrees)
0	7.880	−0.260	313.720
1	8.490	−7.290	332.330
2	7.680	2.820	349.930
3	6.300	4.210	353.180
4	4.580	4.950	359.870
5	2.740	5.010	355.840
6	1.010	4.410	356.300
7	0.259	3.880	3.900
8	−0.400	−3.090	3.670
9	0.250	−3.760	3.690
10	1.000	−4.290	4.150
11	2.730	−4.890	5.120
12	4.560	−4.830	6.810
13	6.280	−4.090	10.000
14	7.660	−2.700	13.000
15	8.440	−0.610	18.000
16	7.790	−2.690	46.270

The procedure converged for all given inital guesses, produced by means of a random number generating subprogram, in less than 50 iterations (usually around 20)Contrary to the determined case (5 prescribed configurations), for which two different meaningful solutions exist, for the cases tried here the procedure converged always to the same single solution, except for 6 and 17 configurations, which produced two different solutions.The error in the approximation was normalized, to yield a dimensionless number, in the following way: Let

$$f_j=|e^{i\theta'_j}(a_0-r_0)+r_j-b|^2-|a_0-b|^2, j=1,\ldots,m \qquad (14)$$

If the synthesis were exact, then all f_j would be negligibly small. In approximate syntheses, however, these functions attain finite values. The kinematic meaning of these values is that they represent the difference between the length of the RR dyad in its initial configuration, and that in its jth configuration, i.e. A_jB-A_0B,if the synthesized linkage were to satisfy the prescribed conditions exactly. The dimensionless error in the approximation, e_j, associated with the jth configuration, is then

$$e_j=|f_j|/|a_0-b|^2, j=1,\ldots,m \qquad (15)$$

where a_0 and b are those obtained from the least-square solution to the nonlinear system of equations. Notice that the errors thus defined are quadratic. To obtain a representative value of the overall error, the average of the square roots of the m errors defined in (14) should be taken, i.e.

$$e=\sum_1^m \sqrt{|f_j|} \;/\; n|a_0-b|$$

Some of the results obtained are shown next.

TABLE 4. Overdetermined synthesis of the RR dyad for rigid-body guidance.

For 6 configurations,	
First solution:	Second solution:
a_0=-0.961467-i2.826960	a_0=7.690390+i7.700030
b^0=-1.643590-i7.997190	b^0=0.748466-i0.609952
Error = 17.02%	Error = 13.49%
For 17 configurations,	
First solution:	Second solution:
a_0=5.123750+i2.254620	a_0=1.443950-i6.704520
b^0=0.549476-i7.03377	b^0=6.370810-i9.315060
Error = 38.74%	Error = 60.77%

CONCLUSIONS

Householder reflections appear to be far more efficient in solving linear problems arising within the field of unconstrained optimal synthesis of linkages. As to nonlinear problems, the extension is straightforward. Regarding constrained problems, these could be handled using this method by introducing suitable slack variables and penalty functions. As to processor times, the first example consumed 11.8 sec, whereas the time reported (8)using Powell's method is 2.2 min, the method introduced here thus appearing to be more economical. With regard to the synthesis for rigid-body guidance, it is necessary to investigate whether for overdetermined problems, in general two different solutions can be expected, thus enabling the designer to synthesize RRRR plane linkages for overdetermined rigid-body guidance problems.

ACKNOWLEDGEMENTS

This research project was sponsored by the School of Engineering at the National Autonomous University of Mexico. The computer programming was performed by Mr. Mario Siller. Householder's method was implemented via subroutines HECOMP and HOLVE, due to Moler (11)

REFERENCES

1. Denavit J. and Hartenberg R.S. Kinematic Synthesis of Linkages, Mc Graw-Hill, N. York, 1964
2. Suh C.H., and Radcliffe C.W., Kinematics and Mechanisms Design, Wiley, N. York, 1978
3. Ben-Israel A. and Greville T.N.E., Generalized Inverses: Theory and Applications, Wiley, N.York, 1974, pp.103-104
4. Stewart G.W., Introduction to Matrix Computations, Academic Press, N. York, 1973,pp.208-249
5. Björk Å. and Dahlquist G., Numerical Methods, Prentice-Hall, Englewood Cliffs, 1974,pp.443-444.
6. Businger P. and Golub G.H., "Linear Least Squares Solutions by Householder Transformations", in Wilkinson J.H. and Reinsch C., eds., Handbook for Automatic Computation, Vol II, Springer-Verlag, N. York, 1971, pp. 111-118
7. Fox R.L and Gupta K.C., "Optimization Technology as Applied to Mechanism Design", Journal of Engineering for Industry,Trans. ASME, Series B.Vol. 95, May 1973, pp. 657-663
8. Suh C.H. and Mecklenburg A.W.,"Optimal Design of Mechanisms with the Use of Matrices and Least Squares", Mechanism and Machine Theory, Vol.8,pp. 479-495
9. Powell M.J.D., "An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives", Computer Journal, vol.7-No.4,1964,pp.303-307
10. Burmester R., Lehrbuch der Kinematic,Vol.1, „Die ebene Bewebung", Verlag von Arthur Felix, 1886
11. Moler C.B., Matrix Eigenvalue and Least-Square Computations, Computer Science Department, Stanford University, March 1973

DIVISION DE EDUCACION CONTINUA

Facultad de Ingeniería, UNAM

OPTIMACION DE SISTEMAS FISICOS

TEMA 5 Problemas Asociados a Sistemas Eléctricos

" DESPACHO ECONOMICO DE CARGA "

DR. EDUARDO ARRIOLA VALDES

MARZO, 1981.

Palacio de Minería Calle de Tacuba 5, primer piso México 1, D.F.

INTRODUCCION

El objetivo primordial en la operación de un sistema eléctrico de potencia es el de proporcionar a los consumidores la energía eléctrica un servicio sin interrupciones a frecuencia constante y dentro de límites tolerables de voltaje en todos los puntos de suministro. Esto implica el controlar en tiempo real la generación para satisfacer en todo instante las contínuas variaciones de la demanda. Esta acción, de igualar minuto a minuto la generación a la demanda, es un problema fundamental en todo sistema de conversión de energía y se toma particularmente complejo en una red eléctrica en donde se operan en forma simultánea un gran número de unidades generadoras de muy diversas características y entre las que se destacan por su importancia las siguientes :

a) Capacidad de generación

b) Tipo de combustible utilizado

c) Eficiencia

d) Costos de operación

e) Características de respuesta

El problema de despacho de carga consiste en decidir que porción de la demanda deberá ser suministrada por cada una de las unidades operando en el sistema y además de las características antes mencionadas deberá considerar los siguientes factores adicionales:

a) Efectos de la distribución de la generación sobre las pérdidas de transmisión.

b) Capacidad de transmisión de las líneas.

c) Políticas de localización de reserva rodante.

d) Acuerdos de compra-venta de energía con otros sistemas.

e) Políticas de operación de plantas hidroeléctricas.

Si al resolver el problema de despacho de carga se busca entre las múltiples soluciones factibles aquella que minimize el costo de operación, el problema se denomina despacho económico. El propósito de este trabajo es el de plantear este último problema y presentar algunas de las técnicas utilizadas en su solución.

CONSIDERACIONES Y DEFINICIONES

Los avances logrados en años recientes en la tecnología de conversión de energía, caracterizados por incrementos en tamaño, presión y temperatura de las unidades generadoras de vapor, han dado como resultado una contínua mejoría en la eficiencia de operación de las unidades! En consecuencia ha aumentado la disparidad en el consumo específico de las fuentes alternativas de generación, que podrían en un tiempo dado ser utilizadas para satisfacer una determinada demanda. Esta disparidad aunada a las variaciones en los costos de combustible y en los factores de pérdidas por transmisión, hacen necesaria la utilización de técnicas eficientes de despacho económico para lograr ahorros considerables de combustible al operar un sistema eléctrico de potencia.

Como un ejemplo de los avances logrados se puede mencionar[1] que en el año de 1940 el consumo específico para generación de energía eléctrica utilizando combustibles fósiles era en promedio de 4132.8 KCal/KW Hr (16,400 BTU/KW Hr). Para 1950 este promedio había mejorado a 3528 KCal/KW hr (14,000 BTU/KW hr) y para 1960 a 2772 KCal/KW hr (11,000 BTU/KW hr). En la tabla 1 se pueden observar los consumos específicos de unidades generadoras de diferente tipo y capacidad. Estos datos fueron obtenidos en un estudio reciente realizado en CFE[2]

TIPO DE COMBUSTIBLE	CAPACIDAD (MW)	CONSUMO ESPECIFICO (KCal/KW hr)
Carbón	600	2260
Carbón	300	2420
Combustoleo	600	2220
Combustoleo	300	2375
Combustoleo	150	2450
Gas	50	3960

Tabla 1. Tipo, Capacidad y Consumos Específicos de Plantas

La efectividad con que una planta térmica cumple su propósito de conversión de energía se define por su curva de entrada-salida, en la que se representa la entrada (H) en el eje de las ordenadas en KCal/hr y en el eje de las abcisas la salida (P) en MW. Una curva típica de entrada-salida se muestra en la figura 1.

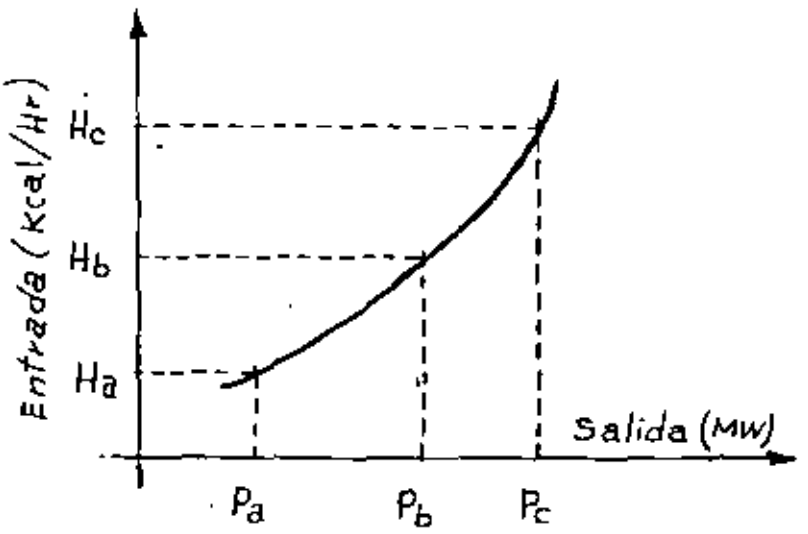


Figura 1. Curva Típica de Entrada-Salida

Para una salida dada el cociente de entrada entre salida recibe el nombre de consumo específico. Este cociente se puede graficar contra la salida como se muestra en la figura 2.

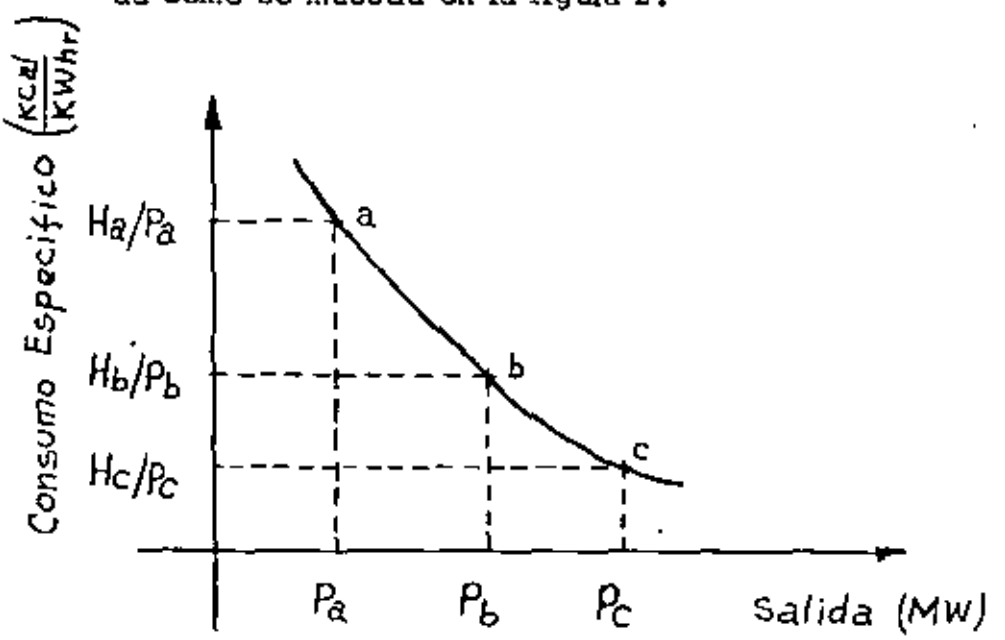


Figura 2. Curva de Consumo Específico

La eficiencia de la unidad generadora se define como el cociente de la salida entre la entrada, de ahí que la eficiencia promedio para una salida dada sea inversamente proporcional al consumo específico.

Para obtener la curva de costo contra potencia generada se multiplica el costo del combustible dado en pesos por kilocaloría por la entrada en kilocaloría/hora y esta curva se representa en la figura 3.

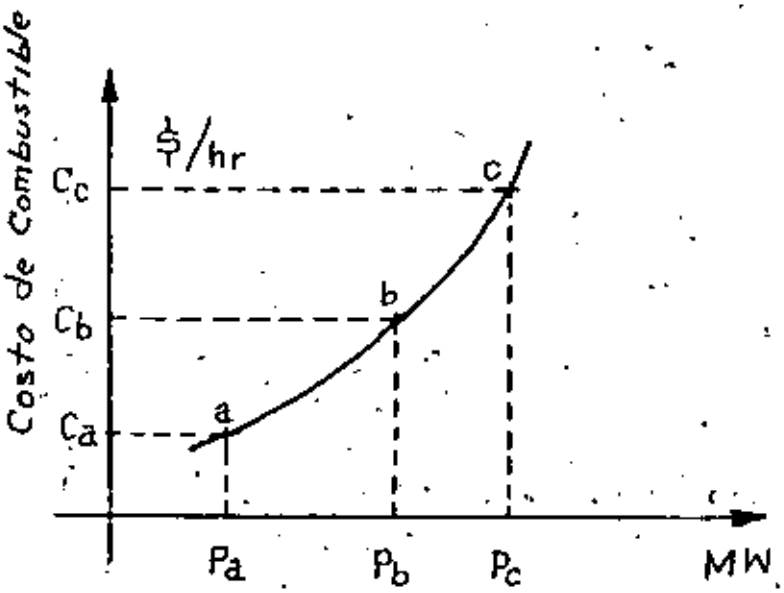


Figura 3. Curva Típica de Costo de Produccion Contra MW

PLANTEAMIENTO DEL PROBLEMA DE DESPACHO ECONOMICO SIN PERDIDAS

Tal como se mencionó con anterioridad, el objetivo del despacho económico es el de satisfacer con costo mínimo una demanda determinada. Matemáticamente el problema se plantea como un problema de optimización en

los siguientes términos:

minimizar : $C_T (PG_i) = \sum_{i=1}^{g} C_i (PG_i)$ (1)

respetando la restricción de satisfacer la demanda :

$$\sum_{i=1}^{g} PG_i = PD \quad (2)$$

y las restricciones de operación de las unidades generadoras

$$PG_i^{min} \leq PG_i \leq PG_i^{max} \quad (3)$$

en donde:

C_T = Costo total de generación

$C_i (PG_i)$ = Costo de generar "PG_i" MW con la i-ésima unidad (Fig. 3).

PG_i = Nivel de generación de la unidad "i"

PD = Demanda total del sistema

PG_i^{min} = Límite inferior de operación de la unidad "i"

$PG_i^{máx}$ = Límite superior de operación de la unidad "i"

g = Número de generadores en el sistema

La técnica básica en una gran cantidad de métodos de optimización es la de convertir el problema con restricciones en un problema de minimización sin restricciones. Esta conversión se logra introduciendo una nueva variable por cada restricción de igualdad en el problema (del tipo de (2)). En nuestro caso tendríamos

minimizar $L(PG_i, \lambda) = C_T (PG_i) + \lambda (PD - \sum_{i=1}^{g} PG_i)$ (4)

en donde la nueva función objetivo es L y el número de variables se ha incrementado a g + 1. Es un hecho conocido que el mínimo de una función como (4) se encuentra en el punto donde las parciales de dicha función con respecto a sus variables es cero, entonces :

$$\frac{\partial L}{\partial PG_i} = 0 \; ; \quad \frac{\partial C_T(PG_i)}{\partial PG_i} = \lambda \qquad (5)$$

$$\frac{\partial L}{\partial \lambda} = 0 \; ; \quad PD = \sum_{i=1}^{g} PG_i \qquad (6)$$

De la ecuación (5) se puede observar que la solución del problema de despacho se encuentra donde todos los costos incrementales de generación $\partial C / \partial PG$ son iguales y la ecuación (6) indica que esta solución cumple la restricción de satisfacer la demanda PD.

Un algoritmo para encontrar la solución al problema de despacho, simplificado al no tomar en consideración las pérdidas de transmisión, sería el siguiente:

1. Da un valor inicial a la variable λ
2. Encuentra el valor de cada PG_i de la ecuación 5
3. Si el valor encontrado de PG_i excede el límite superior entonces $PG_i = PG_i^{max}$; si el valor encontrado en PG_i es menor que el límite inferior entonces $PG_i = PG_i^{min}$. Esta verificación se hace para cada una de las unidades generadoras.
4. Si $\sum_{i=1}^{g} PG_i = PD$ termina el proceso . En caso contrario continúa en el paso 5.

5. Si $\sum_{i=1}^{g} PG_i > PD$ reduce el valor de λ.

   Si $\sum_{i=1}^{g} PG_i < PD$ incrementa el valor de λ.

6. Con el nuevo valor de λ regresa al punto 2.

Como se mencionó antes este algoritmo no toma en consideración las pérdidas de transmisión sin embargo, se incluyó por su sencillez que a la vez ilustra el mecanismo para obtener una distribución económica de la generación que satisface una demanda dada.

PLANTEAMIENTO DEL PROBLEMA DE DESPACHO ECONOMICO CON PERDIDAS DE TRANSMISION

Es posible a través de un conjunto de constantes conocidas como constantes B incluir las pérdidas de transmisión sin necesidad de conocer en el instante de cálculo la verdadera topología del sistema. Obviamente el método da resultados aproximados pero es un paso adelante del método anterior en donde las pérdidas son totalmente ignoradas. El planteamiento contiene ligeras variantes del anterior como se puede observar:

Minimizar $$C_T (PG_i) = \sum_{i=1}^{g} C_i (PG_i) \qquad (7)$$

sujeto a : $$\sum_{i=1}^{g} PG_i = PD + PL (PG_i) \qquad (8)$$

y $$PG_i^{min} \leq PG_i \leq PG_i^{max} \qquad (9)$$

en donde $P_L (PG_i)$ es función de las pérdidas y esta dada por :

$$PL\,(PG_i) = B_0 + \sum_{i=1}^{g} B_i PG_i + \sum_{i=1}^{g} \sum_{j=1}^{g} PG_i\, B_{ij}\, PG_j \qquad (10)$$

y las B's que intervienen en esta fórmula de pérdidas son constantes calculadas por fuera del proceso. El método de cálculo de estas constantes sale del objetivo de este trabajo y se puede estudiar en las referencias 3 y 4.

Nuevamente la técnica de solución consiste en minimizar una función objetivo a la que se han agregado las restricciones de igualdad, esto es:

$$\text{minimizar } L(PG_i, \lambda) = C_T(PG_i) + \lambda\,(PD + PL - \sum_{i=1}^{g} PG_i) \qquad (11)$$

Al igualar a cero las parciales de L con respecto a sus variables obtenemos el conjunto de ecuaciones.

$$\frac{\partial L}{\partial PG_i} = \frac{\partial C_T}{\partial PG_i} + \lambda\left(\frac{\partial PL}{\partial PG_i} - 1\right) = 0 \qquad (12)$$

$$\text{y} \quad \frac{\partial L}{\partial \lambda} = PD + PL - \sum_{i=1}^{g} PG_i = 0 \qquad (13)$$

$$\text{donde} \quad \frac{\partial PL}{\partial PG_i} = B_i + 2 \sum_{j=1}^{g} B_{ij}\, PG_j \qquad (14)$$

La ecuación 12 se puede re-escribir en la forma

$$\frac{1}{\left(1 - \frac{\partial PL}{\partial PG_i}\right)} \frac{\partial C_T}{\partial PG_i} = \lambda \qquad (15)$$

Si comparamos esta ecuación con la ecuación 5 podemos observar que el efecto de las pérdidas de transmisión es el de ajustar los costos incrementales de cada unidad y que en la solución dichos costos incrementales

ajustados son todos iguales. El algoritmo para resolver el problema de despacho económico introduciendo el efecto de las pérdidas de transmisión a través de las constantes B es el siguiente:

1. Dar un valor inicial a λ

2. Resuelve la ecuación 15 para obtener los valores de todas las PG.

3. Compara el valor obtenido para cada PG_i con sus límites de operación y :

   si $PG_i > PG_i^{max} \rightarrow PG_i = PG_i^{max}$

   si $PG_i < PG_i^{min} \rightarrow PG_i = PG_i^{min}$

4. Calcula de la ecuación 10 el valor de las pérdidas PL.

5. Si: $\sum_{i=1}^{g} PG_i = PD + PL$ termina; en caso contrario continúa en el paso 6.

6. Si $\sum_{i=1}^{g} PG_i > PD + PL$ reduce al valor de λ.

   Si $\sum_{i=1}^{g} PG_i < PD + PL$ incrementa el valor de λ.

7. Con el nuevo valor de λ regresa al paso 2.

El siguiente grado de complejidad en la solución del problema de despacho económico consiste en incluir la solución de la red para considerar de manera exacta las pérdidas de transmisión. Obviamente el uso de esta técnica, denominada flujos óptimos por incluir la solución del problema de flujos de carga en el proceso de optimización, requiere de mayor información que las técnicas anteriores a saber, la topología del sistema, los paráme

tros de las líneas y las inyecciones de potencia en los diferentes nodos de la red.

FLUJOS OPTIMOS EN LA SOLUCION DEL PROBLEMA DE DESPACHO ECONOMICO

Los flujos de potencia en un sistema de N nodos se caracterizan por el conjunto de ecuaciones complejas

$$S_k^* = V_k \angle -\delta_k \sum_{m=1}^{N} \left(G_{km} + j B_{km} \right) V_m \angle \delta_m \qquad (16)$$

en donde

V_k : magnitud de voltaje en el nodo k'

δ_k : ángulo de fase en el nodo k

$G_{km} + j B_{km}$: elemento de la matriz de admitancia nodal

$S_k^* = P_k^N - j Q_k^N$: potencia compleja neta inyectada en el nodo "k"

Utilizando la notación $P_k (V, \delta) - j Q_k (V, \delta)$ para el lado derecho, la ecuación 16 puede escribirse como un conjunto de 2N ecuaciones del tipo

$$P_k^N - P_k (V, \delta) = 0, k = 1, \ldots, N \qquad (17)$$

$$Q_k^N - Q_k (V, \delta) = 0, k = 1, \ldots, N \qquad (18)$$

Como es sabido cada nodo de la red está caracterizado por cuatro variables, P_k^N, Q_k^N V_k y δ_k y en el problema de flujos se especifican dos de ellas mientras las otras dos son incógnitas. Dependiendo de que variables

sean especificados los nodos de la red se pueden clasificar en tres tipos como se muestra en la tabla 2.

NODO TIPO	DATOS	INCOGNITAS
Referencia	V, δ	P^N , Q^N
Carga	P^N , Q^N	V , δ
Voltaje controlado	V, P^N	δ , Q^N

Tabla 2. Diferentes Tipos de Nodo en el Problema de Flujos

Estas variables pueden clasificarse como: variables dependientes (x). variables de control (u) y parámetros (p), esto es :

$$[x] = \begin{bmatrix} \cdot V \\ \cdot \delta \\ \cdot \delta \end{bmatrix} \quad \begin{matrix} \\ \text{en c/u de los nodos de carga} \\ \\ \text{en c/u de los nodos de v. controlado} \end{matrix}$$

$$[u] = \begin{bmatrix} \cdot V & : & \text{en el nodo de referencia} \\ \cdot V & : & \text{en c/u de los nodos de voltaje controlado} \\ \cdot PG & : & \text{potencia de generación disponibles para despacho económico} \\ \cdot \text{etc.} & & \end{bmatrix}$$

$$[p] = \begin{bmatrix} P^N , Q^N & \text{en nodos de carga} \\ \delta & : \text{en el nodo de referencia} \\ \cdot \text{parámetros de líneas} & \\ \cdot \text{etc.} & \end{bmatrix}$$

Si definimos en términos de x, u, p, el conjunto de ecuaciones de flujos tenemos :

$$\left[g\ (x,\ u,\ p)\right] = \begin{cases} \text{eq 17} \\ \text{eq 18} \end{cases} \text{para c/u de los nodos de carga.} \qquad (19)$$

eq 17 para cada uno de los nodos de v. controlado

podemos entonces plantear el problema de flujos óptimos de la siguiente manera

minimiza $C_T(x,u)$(Costo total de Gen.,) (20)

Sujeto a las ecuaciones de la red

$$g\ (x,\ u,\ p) \qquad (21)$$

y a los límites de operación de las variables de control

$$U^{min} \leq U \leq U^{max} \qquad (22)$$

Nuevamente al igual que en los casos anteriores introducimos tantas variables auxiliares como restricciones de igualdad haya en la ecuación (21) y convertimos el problema a un problema de minimización sin restricciones esto es :

minimiza $L\ (x,\ u,\ \lambda\ ) = C\ (x,u) + \lambda^T g(x,\ u,\ p)$ (23)

Las condiciones necesarias para el mínimo de (23) estarán dadas por las siguientes ecuaciones:

$$\frac{\partial L}{\partial x} = \frac{\partial C_T}{\partial x} + \frac{\partial g}{\partial x}^T \lambda = 0 \qquad (24)$$

$$\frac{\partial L}{\partial u} = \frac{\partial C_T}{\partial u} + \frac{\partial g}{\partial u}^T \lambda = 0$$

$$\frac{\partial L}{\partial \lambda} = g(x, u, p) = 0 \qquad (25)$$

en donde el último conjunto de ecuaciones es nuevamente el conjunto de ecuaciones de flujos dado por (19).

Las ecuaciones (24), (25) y (19) son no-lineales y su solución debe obtenerse en forma iterativa. El método más simple para obtener la solución es el llamado método del gradiente cuya idea básica es la de pasar de una solución factible a otra moviéndose en la dirección contraria del gradiente ya que este vector apunta en la dirección en que la función objetivo L tiene mayor crecimiento (en nuestro problema el gradiente está dado por la ecuación 25). El algoritmo básico para la solución de este problema es el siguiente:

1. Dar un valor inicial a las variables de control u
2. Con los valores presentes de u resuelve el conjunto de ecuaciones no lineales (19)
3. Encuentra el valor de λ resolviendo la ecuación
$$\lambda = - \frac{\partial g}{\partial x}^{T-1} \frac{\partial C_T}{\partial x}$$
4. Con el valor de λ calcula el gradiente
$$\nabla L = \frac{\partial C_T}{\partial u} + \frac{\partial g}{\partial u} \lambda$$
5. Si el gradiente es lo suficientemente pequeño, termina el proceso, si no continúa en el paso 6.

6. Encuentra un nuevo conjunto de variables de control

$$u^{\text{nueva}} = u^{\text{anterior}} + \Delta u$$

$$y \quad \Delta u = -\alpha \nabla L$$

7. Verifica que los nuevos valores de u no violan las restricciones y si:

$$u_i^{\text{nuevo}} > u_i^{\max} \rightarrow u_i^{\text{nuevo}} = u_i^{\max}$$

$$u_i^{\text{nuevo}} < u_i^{\min} \rightarrow u_i^{\text{nuevo}} = u_i^{\min}$$

8. Con los nuevos valores de u encontrados en los pasos 6 y 7 regresa al paso 2.

En el paso 6 del algoritmo que es en el que se calculan los cambios al vector de control se introdujo una nueva variable α. Esta variable tiene la función de graduar el cambio para evitar problemas de oscilaciones alrededor del mínimo y existen técnicas para seleccionarla óptimamente[5] en cada ciclo de ajuste de las variables de control.

CONCLUSIONES

La diversidad en tipos, eficiencia y tamaños de las plantas termoeléctricas y los altos costos que los combustibles fósiles han alcanzado en los últimos años, han originado la necesidad de contar con técnicas de despacho de carga que permitan satisfacer la demanda a un mínimo costo. Los avances tecnológicos recientes que han dado pauta a la utilización de las computadoras digitales para el control en tiempo real de los sistemas de potencia han permitido la utilización de técnicas de despacho más sofisticadas que permiten una representación más exacta de las condiciones bajo las que opera el sistema y en consecuencia las posibilidades de obtener mayores

ahorros en la operación aumentan considerablemente.

REFERENCIAS

1. N. Cohn, "Control of Interconnected Power Systems", capítulo 17 "Handbook of Automation Computation and Control," Vol. 3. John Wiley, 1961.
2. Estudio AJBC0-DLP-7730, Oficina de Estudios Especiales de la Gerencia de Estudios e Ingeniería Preliminar, CFE "Base de Costos para los Estudios de Expansión de la Generación", Enero 1977.
3. L.K.Kirchmayer, "Economic Operation of Power Systems", John Wiley, 1958.
4. A.M. Sasson, "Optimal Load Flow a Practical Outlook," en Application of Optimization Methods in Power System Engineering, IEEE Tutorial Course Text, No. 76 CH 1107 - 2 - PNR, 1976.
5. H.W. Dommel y W.F. Tinney. "Optimal Power Flow Solutions", IEEE Trans., Vol. PAS 87, No. 10 Octubre de 1968.
6. O.I. Elgerd, "Electric Energy Systems Theory: An Introduction", Capítulo 8, Mc. Graw Hill, 1971.
7. W.D. Stevenson Jr., "Elements of Power System Analysis" Capítulo 11, Mc. Graw Hill, 1962 (segunda edición).

'alo

DIRECTORIO DE ASISTENTES CURSO OPTIMACION DE SISTEMAS FISICOS (DEL 2 AL 6 DE MARZO DE 1981

NOMBRE Y DIRECCION	EMPRESA Y DIRECCION
1. GUILLERMO JOSE AGUIRRE ESPONDA Guillermo Prieto No. 9 Lomas Quebradas México 20, D. F. Tel: 5-95-12-61	FACULTAD DE INGENIERIA, UNAM Profesor Area de Diseño de Máquinas Ciudad Universitaria México 20, D. F. Tel: 5-50-00-41
2. NICOLAS CALVA TAPIA Nicolas Bravo 17-1 Col. Martín Carrera México 14, D. F. Tel: 5-77-84-39	ENEP-ARAGON-UNAM Profesor Av. Central y Rancho Seco Edo. de México
3. MARCO A. GALINDO SARMIENTO Calle 27 No. 707 Lomas de C.B. Querétaro, Qro.	COMPACTO, S.A. Investigación Libramiento a San Luis Potosí Querétaro, Qro. Tel: 2-14-23
4. JUAN FRANCISCO GARCIA ORDAZ Norte 89 "A" No. 485 Col. Electricistas México 16, D. F. Tel: 5-61-17-21	FACULTAD DE INGENIERIA, UNAM Ayudante de Profesor Ciudad Universitaria México 20, D. F. Tel: 5-50-00-41
5. JUAN DE LA CRUZ HERNANDEZ ZAMUDIO Selene 15 Valle Ensueños Cuautitlan Izcalli, Edo de México	F.E.S.C. (UNAM) Profesor Tiempo Completo (Ingeniería) Cuautitlan de R.R. Campo 4 Cuautitlan, Edo de México
6. RAYMUNDO HUGO RANGEL GUTIERREZ Copildo 300 Edif. 7 Dpto. 1 Copilco Universidad México 21, D. F.	FACULTAD DE INGENIERIA-DIME-DPTO. COMPUTACION Profesor Asociado "A" T.C. Ciudad Universitaria México 20, D. F. Tel: 5-50-15-52 Ext. 3750
7. FELIPE DE JESUS RICALDE RODRIGUEZ Av. Ing. Basiliso Romo Anguiano 12-3 Col. Industrial México 14, D. F. Tel: 5-37-40-75	PROMOCION Y OPERACION, S.A. Soporte Técnico Av. Dr. Andrade No. 60 Col. Doctores México 7, D. F. Tel: 5-88-29-00 Ext. 130

DIRECTORIO DE ASISTENTES CURSO OPTIMACION DE SISTEMAS FISICOS (DEL 2 AL 6 DE MARZO DE 1981

NOMBRE Y DIRECCION	EMPRESA Y DIRECCION
8. EDUARDO SALAS CORDOVA Diana 52 Col. Ensueños Cuautitlan, Izcalli Edo de México	UNAM FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN Maestro Carretera Cuautitlan Teoloyucan Km. 3 Cuautitlan Izcalli, Edo de México
9. JOSE ANTONIO SANCHEZ GUTIERREZ Cipres No. 17 San Juan Cuautitlán, Edo. de México	FACULTAD DE ESTUDIOS SUPERIORES-CUAUTITLAN Profesor Tiempo Completo (Ingeniería) Cuautitlán de R. R. Edo de México
10. ROBERTO VAZQUEZ ZEPEDA Londres 52-201 Col. El Carmen México 21, D. F. Tel: 5-44-83-29	COMISION FEDERAL DE ELECTRICIDAD Supervisor Diseño de Planta Ródano No. 14 Col. Cuauhtémoc México 5, D. F. Tel: 5-53-71-33 Ext. 2098