Staff Papers Series

Staff Paper P78-11 July 1978

AN INTRODUCTION TO ITERATIVE METHODS FOR SOLVING SIMULTANEOUS EQUATION MODELS: ILLUSTRATIONS FROM ECONOMICS

Richard Todd and Terry Roe

Department of Agricultural and Applied Economics

University of Minnesota
Institute of Agriculture, Forestry and Home Economics
St. Paul, Minnesota 55108

AN INTRODUCTION TO ITERATIVE METHODS FOR SOLVING SIMULTANEOUS EQUATION MODELS: ILLUSTRATIONS FROM ECONOMICS

by

Richard Todd and Terry Roe

The assistance of Dr. Kenneth Brown, Department of Computer Sciences, Shih-ta Lin, a Ph.D candidate in the Department of Computer Sciences, Dr. Jean-Claude Koeune, Department of Economics and Dr. Robert Tardiff, Department of Mathematics, University of Minnesota at Morris, were instrumental in earlier drafts of this paper. This effort was supported in part by a 211 U.S. AID grant.

TABLE OF CONTENTS

INTRODUCTION

This paper focuses on the application of computer-based, iterative methods for deriving analytical solutions to economic models expressed as systems of simultaneous linear and/or nonlinear equations. Practical suggestions for using four specific methods are emphasized. Analogies from economics are used to explain practical and theoretical points.

The primary motivation for this paper arises from methodological constraints faced in developing quantitative tools for economic planning, especially in the less developed countries (LDCs), where planners increasingly seek to simulate economic activity at the industrial, sectoral, or national level. Simultaneous-equation models* appear to offer several advantages over other techniques such as input-output analysis or constrained optimization. For instance, input-output analysis is not designed to capture supply and demand responses to substitution possibilities. Constrained optimization models, despite recent innovations [5,6,16], remain limited by the number of equations nonlinear in three or more variables that can be conveniently handled and by convexity conditions required to find an optimal activity set. Simultaneous-equation models are also generally superior for characterizing time-dependent relationships or forecasting macroeconomic activity.

A major impediment to the use of simultaneous-equation models is the difficulty of calculating solutions. This is especially true of large models and models with many nonlinear equations. Procedures used to calculate solutions have been, for the most part, either variations of Gauss-Seidel iteration or _ad hoc_, model-specific methods. In addition to drawing scarce

*That is, quantified models consisting of an equal number of endogenous variables and equations. These models may be termed "nonoptimizing" because their solutions need not depend on the optimization of an explicitly-stated objective function. Some examples are the Kelley-Williamson-Cheetam model [15], the Chenery-Raduchel model [1], and the Wharton and Brookings models [11].

resources away from model specification and estimation, model-specific methods tend to be inflexible. Modifying the model to simulate the effects of policy alternatives can require extensive reformulations of the solution algorithm -- a costly distraction in LDC.planning. The Gauss-Seidel and Jacobi methods used at the Wharton School, Brookings Institute and U.S. Department of Agriculture are generally not as sensitive as model-specific methods to changes in a few equations. However, with Gauss-Seidel methods considerable time and effort may be expended in rearranging the equations of the model before a solution is obtained.

Despite these problems, economists appear to have limited their search for and reporting of methods for solving simultaneous-equation models. For instance, Shapiro and Halabuk's recent review of macroeconomic model building [9] excluded discussion of solution procedures. The general objective of this paper is to partially redress this failing by providing a convenient introduction to iterative methods as solution procedures for simultaneous equation economic models. The theory of iteration and practical suggestions for using iterative methods are presented at a basic level requiring minimal proficiency in mathematics. Part I begins with a very general discussion of the nature of iteration, followed by an optional section which gives a more mathematically detailed treatment. Parts II and III describe four specific iterative methods useful in economic modeling. Suggestions on how to use them are drawn from the literature and from the authors' model solving experience. Part IV summarizes the advantages of the four methods and suggests how to search for other algorithms.

NOTATION

i, j	Indices used as subscripts to indicate the i^{th}, j^{th}, or ij^{th} member of a set. Unless otherwise indicated, i and j take on successive integer values between 1 and n.
$(k-1), (k), (k+1)$, etc.	Indices used as superscripts to indicate the $k-1^{st}$, k^{th}, or $k+1^{st}$ member of a sequence of iterates. Superscript values are nonnegative intergers, $k = 0,1,2,\ldots$.
n	Index used to indicate the total number of equations in a system of equations or the number of elements in a vector or a matrix.
$a,b,\ldots,x,y,z$	Lower case letters denote a real number constant or a variable that takes on real number values. Unless otherwise indicated, variables will be denoted by letters from the end of the alphabet.
$A,B,\ldots,X,Y,Z$	Upper case letters denote a vector or matrix whose elements--a_{ij}, $b_{ij},\ldots,x_{ij}$, y_{ij}, z_{ij} -- may be real constants or variables. Unless otherwise indicated, vectors or matrices with variable elements will be denoted by letters from the end of the alphabet. All matrices used in this paper are square (number of rows = number of columns = n).

$\underline{0}$	A vector or matrix all of whose elements are zero.
$f(x)$	A function of one variable.
$f_i(X)$; or $f_i(x_1,x_2,\ldots,x_n)$	The ith function in a system of n functions in n variables.
$F(X)$	A system of n functions in n unknowns. Collectively, this system of equations is also referred to as a function or a mapping.
$X^{(0)}$	A vector of values which are a user's initial guess at a solution of a system of simultaneous equations.
$\ldots,X^{(k-1)},X^{(k)},X^{(k+1)},\ldots$	Vectors representing the k-1st, kth, and k+1st iterates in a sequence of iterates generated by an iterative method beginning at some $X^{(0)}$ and searching for the solution of a system of simultaneous equations.
$x_i^{(k)}$	The ith element of the vector $X^{(k)}$.
$x^{(0)},\ldots,x^{(k)}$	The sequence of iterates generated by a one-dimensional iterative method beginning at $x^{(0)}$.
$X^{(k+1)}=G(X^{(k)})$	A function that expresses new iterates, $X^{(k+1)}$, in terms of old iterates, $X^{(k)}$.
$x_i^{(k+1)}=g_i(X^{(k)})$; or $x_i^{(k+1)}=g_i(x_i^{(k)}x_2^{(k)},\ldots,x_n^{(k)})$	The ith function in the system of n iterative functions collectively denoted $X^{(k+1)}=G(X^{(k)})$.
J; or $J(X)$	The Jacobian matrix of a system of equations, evaluated at X.

$\rho(X)$	The spectral radius of matrix X.
$\lvert x \rvert$	The absolute value of x.
$\lvert\lvert X \rvert\rvert$	The norm of the vector or matrix X.
*	A superscript that denotes a point x^* or X^* that is an economic equilibrium, the solution of a system of simultaneous equations, or the limiting point of a sequence of iterates.
$f'(x)$	The derivative, $\frac{df(x)}{dx}$, of a function of one variable, evaluated at x.
$\frac{\partial f(x)}{\partial x_j}$	The j^{th} first partial derivative of a function of n variables, evaluated at X.
$\frac{\partial f_i(X)}{\partial x_j}$	The j^{th} first partial derivative of the i^{th} equation in a system of n equations, evaluated at X. Thus, $\frac{\partial f_i(X)}{\partial xj}$ can represent the ij^{th} element of the Jacobian matrix of the system of equations.
$\sum_{j}$	The summation of a sequence of terms containing variables with the subscript j. Unless otherwise indicated, the sum shall be formed by taking integer values of j between 1 and n.
$\sum_{j \leq (i-1)}$	Same as above except that the index j takes on integer values from 1 to i-1, where i is fixed.
$\sum_{j \leq (i+1)}$	Same as above except that the index j takes on integer values from i+1 to n, where i is fixed.

$\sum_{j \neq i}$	Same as above except that j takes on all integer values from 1 to n except i, where i if fixed.

Part I: The Nature of Iteration

A. Basic Concepts.

Iteration is a method for finding a solution to a problem by trial and error. The method begins with an initial guess at a solution. This guess is then tested to see if it is an adequate answer to the problem. If it is not, a second guess is generated by modifying the initial guess according to a well-defined rule. The second guess is tested; and, if need be, a third guess is generated by modifying the second guess according to the same rule. This process continues until a solution is found or until a decision is made to stop looking.

There are many possible rules for generating new iterative guesses from old ones, and each distinct rule defines a distinct iterative method. Most rules allow information about the problem-to-be-solved to influence the way old guesses are modified to form new guesses. Each new guess thus depends on the rule, the problem, and the previous guesses. However, for a given rule and problem, new guesses depend only on old guesses.

In the case of iterative solution of simultaneous-equation economic models, the following components of the iterative process merit identification:

1. The problem is to find a solution for an economic model expressed as a system of n equations in n "unknowns" (endogenous variables). In functional notation, $F(X) = \underline{0}$ is the model, where F is a function, X is an nx1 vector of endogenous variables, and $\underline{0}$ is an nx1 vector of constants, $X^{(*)}$, which satisfies the model. I.e., $F(X^{(*)}) = \underline{0}$.
2. The guesses, or iterates, are nx1 vectors of constants to be tested as possible solutions of the model. They are generated and tested one at a time, beginning with $X^{(0)}$, the initial guess, and proceeding

through subsequent guesses $X^{(1)}$, $X^{(2)}$, $X^{(3)},\ldots,X^{(k)}$, $X^{(k+1)}$, etc., where superscripts indicate the ordering of the sequence of guesses.

3. A test determines whether the most recent guess is an adequate approximation to a solution. It consists of measuring (by means of some distance formula, D) the deviation of $F(X^{(k)})$ from zero. I.e., the quantity $D[F(X^{(k)}) - \underline{0}] = D[F(X^{(k)}) - F(X^{(*)})]$ must be approximately zero for $X^{(k)}$ to be accepted as a solution to the model. Otherwise a new iterate, $X^{(k+1)}$, must be chosen.

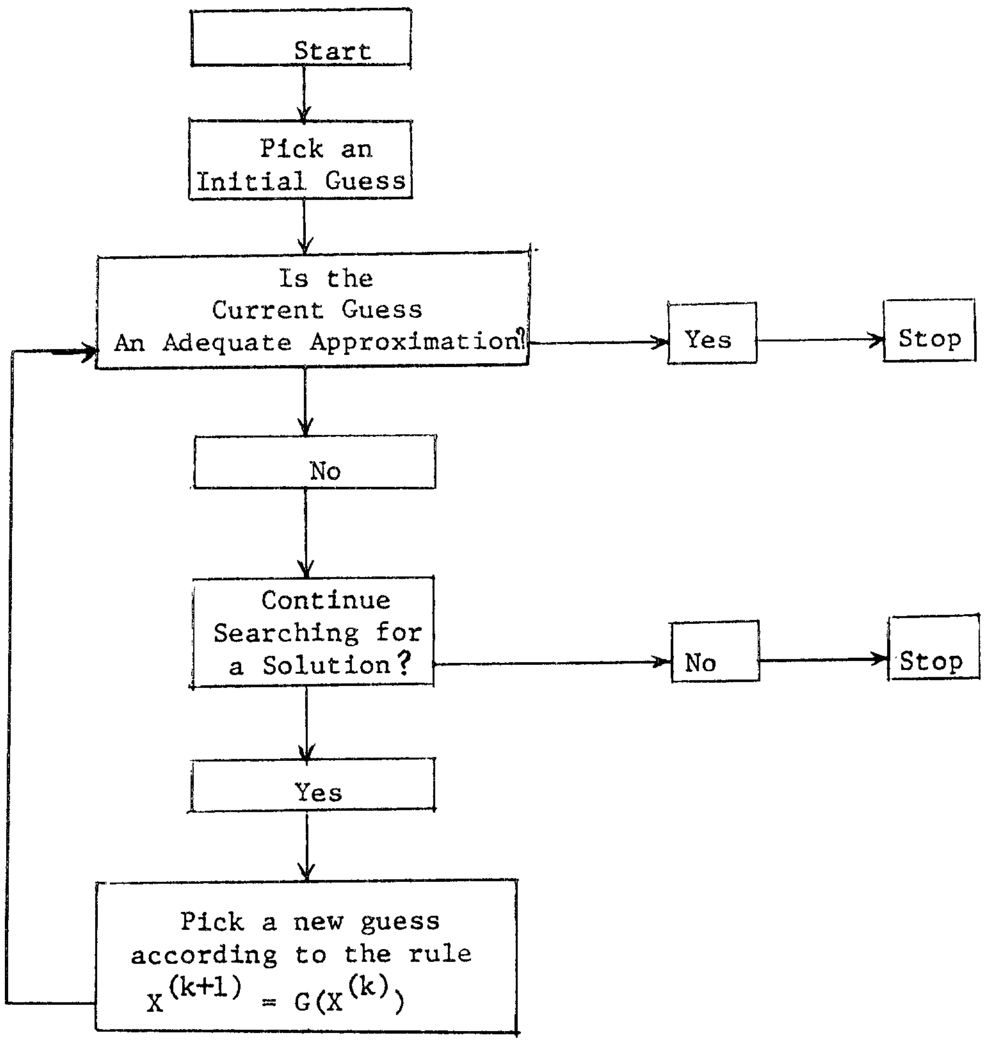
4. The rule generates new guesses from old guesses. Each application of the rule to derive a new guess is called an iteration. A rule can be written as the function[1/] $X^{(k+1)} = G(X^{(k)})$, where G incorporates procedures for obtaining "information" about the model's "behavior" near the point $F(X^{(k)})$ and for using this information in the selection of $X^{(k+1)}$.

Figure 1 is a flowchart of the workings of an iterative method.

If an iterative method generates a sequence of iterates that terminates in values arbitrarily close to a solution[2/], the method is said to converge to a solution, or to be convergent. Methods which converge to a solution from any initial guess are globally convergent. For some nolinear models, globally convergent iterative methods do not exist; it is only possible to find methods that converge when the initial guess lies in some "region of convergence" surrounding a solution. These methods are locally convergent.

In practice, the same method may be globally convergent when applied to a certain class of models, and only locally convergent or not convergent at all for other models. Convergence may also depend on the way a given model is expressed algebraically to address the computer program for the iterative method. A researcher seeking solutions to a model faces the twin problems

Figure 1: Schematic Structure of an Iterative Method


Start
Pick an Initial Guess
Is the Current Guess An Adequate Approximation?
Yes
Stop
No
Continue Searching for a Solution?
No
Stop
Yes
Pick a new guess according to the rule $X^{(k+1)} = G(X^{(k)})$

of (1) choosing an iterative method appropriate to his model, and (2) choosing an algebraic statement of his model appropriate to the iterative method.

Hundreds of iterative methods have been proposed for solving systems of simultaneous equations. The four discussed in Parts II and III are (1) theoretically fundamental, (2) easy to program and use, (3) efficient and reliable, and (4) applicable to both linear and nonlinear models. The Jacobi and Gauss-Seidel methods presented in Part II were originally devised for linear problems. Newton's method and the related Brown's method are discussed in Part III. The two pairs of methods are described and suggestions are made on how to modify either the model or the methods in order to obtain convergence. Readers may wish to proceed directly to Parts II or III. The remainder of Part I is an optional section which introduces some concepts and theorems important in the theory of iterative convergence. It provides additional background for interested readers but is not essential for understanding Parts II-IV.

B. A Sketch of the Theory of Convergence of Iterative Methods

Consider the iterative process defined over some domain D by the function $X^{(k+1)} = G(X^{(k)})$. (Note that the function G depends both on some model, $F(X) = 0$, and on the particular iterative technique used to solve the model. Neither $G(X^{(k)}) = X^{k+1}$ nor any of its derivatives should be confused with $F(X) = 0$ and its derivatives.) The method converges to X^* if as k gets arbitrarily large the distance $|x_i^* - x_i^{(k)}|$ tends to zero for each element. When, for sufficiently large k, $|x_i^* - x_i^{(k)}|$ and hence $|x_i^* - x_i^{(k+1)}|$ approach zero, the successive iterates $X^{(k)}$ and $X^{(k+1)}$ are approximately equal since $x_i^{(k)} \simeq x_i^* \simeq x_i^{(k+1)}$, or $x_i^{(k)} \simeq x_i^{(k+1)}$. Thus, X^* is a fixed point of the iterative method as well as a solution to the economic model,[3] and an alternative criterion of convergence is $|x_i^{(k)} - x_i^{(k+1)}| \simeq 0$.

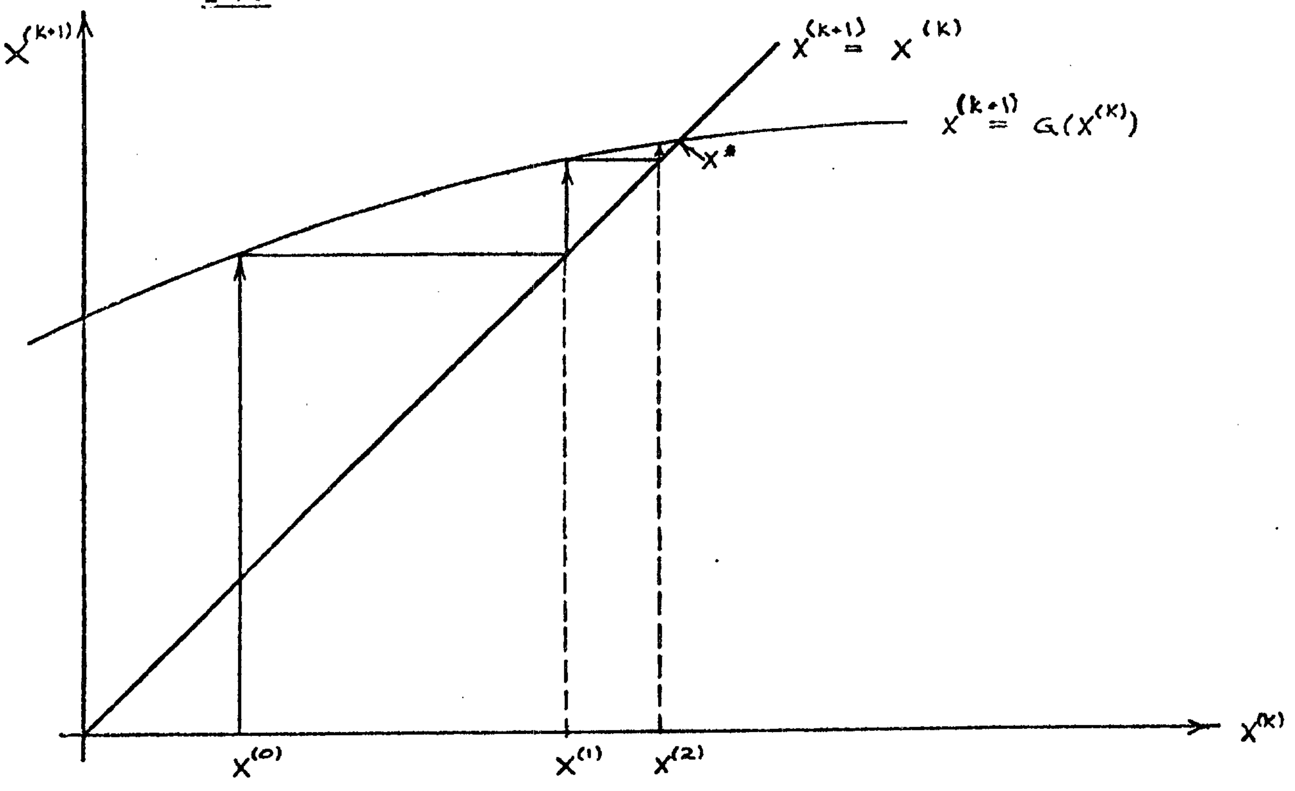
The analysis of convergence in a one-equation problem can be presented graphically. Each of the following four graphs (derived from /4, p. 3/) illustrates the effects of the slope of the iterative function, $g(x^{(k)})$, on the convergence of the method.

In 2A and 2B the sequence of iterates begins at the initial guess $x^{(0)}$ and converges to the fixed point x^*. In 2C and 2D the sequence diverges from x^* despite relatively good initial guesses. The slope of $g(x^{(k)})$, which is shallow in 2A and 2B but relatively steep in 2C and 2D, determines the convergence or divergence of the sequence. Indeed, the sequence will always converge if the initial guess lies in a neighborhood of the solution where $g(x^{(k)})$ is less steep than the line $x^{(k+1)} = x^{(k)}$. This line passes through the origin at an angle of 45° and has a slope of +1. It follows that an iterative method will converge to x^* from any $x^{(0)}$ in a neighborhood of x^* where the slope of $g(x^{(k)})$ is less than 1 in absolute value, i.e., where $|g'(x^{(k)})|<1$.[4/] This result should come as no surprise to agricultural economists. The same sort of condition (slope of demand curve steeper than slope of supply curve)[5/] determines the convergence or divergence of a cobweb model. (This analogy is more fully developed below in the explanation of the Gauss-Seidel method, and also in Appendix B.)

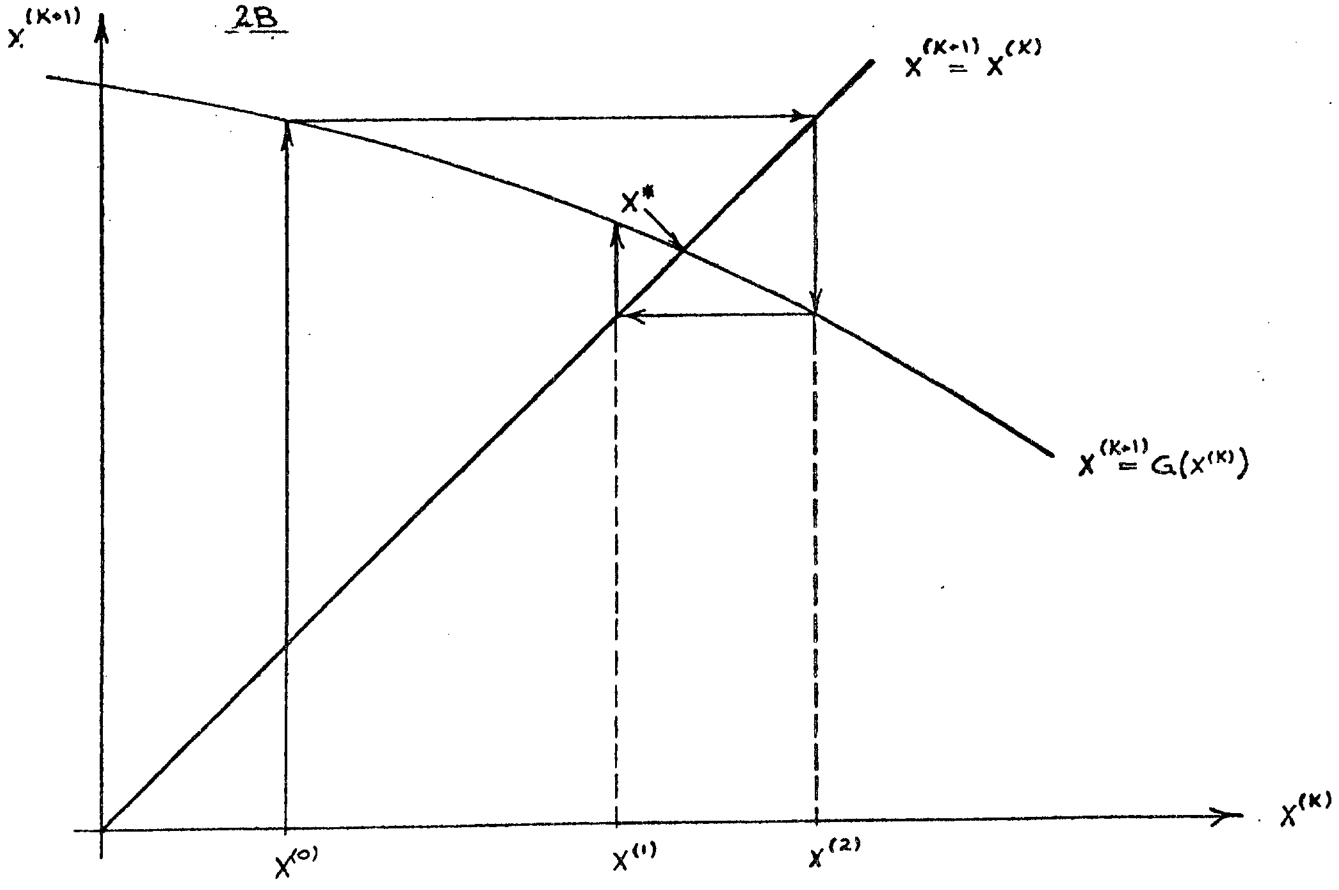
In order to better understand the nature of convergence of iterative methods in more realistic models, these simple results must be generalized to higher dimensions. The theorems that follow show that when the matrix of first partial derivatives of the n-dimensional iterative function G(X) has a "magnitude" which is less than one throughout a convex neighborhood of the solution, a sequence of iterates in that neighborhood will have decreasing first differences and will converge to the solution X^*. To state this

FIGURE 2

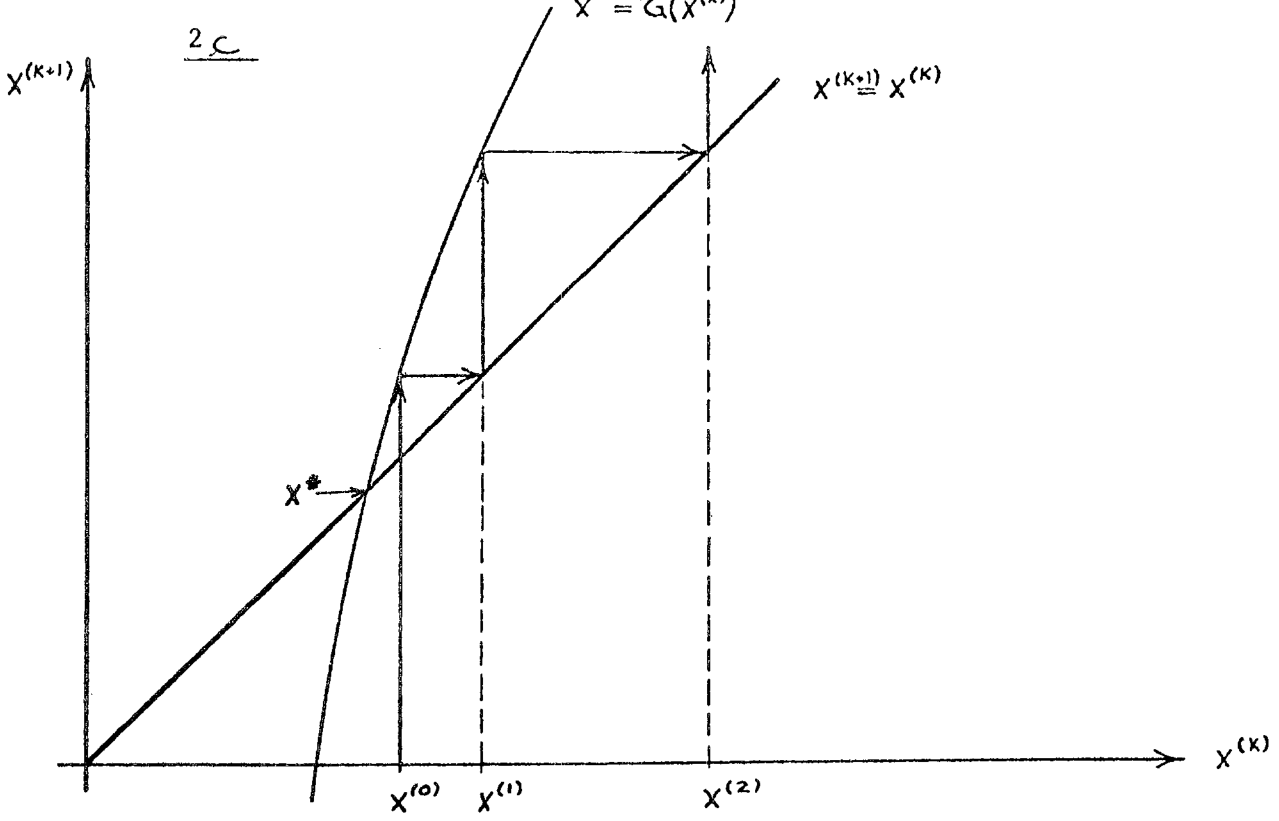
2.A



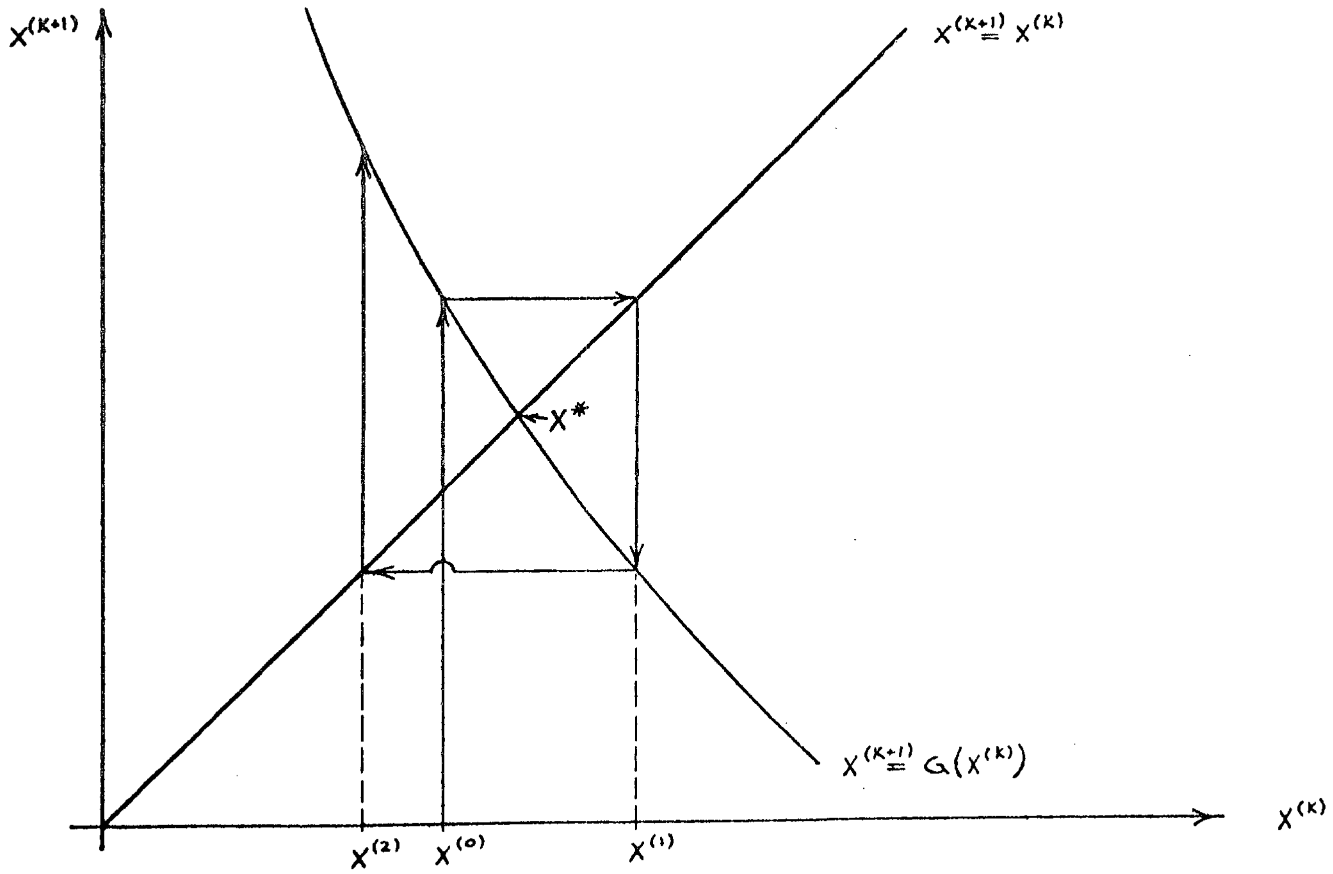
2B



2C



2D



result formally, the concepts of matrix and vector norms, Jacobian matrices, spectral radii, and contraction mappings are introduced next.

1. Vector and Matrix Norms. For real numbers a measure of magnitude is the absolute value, signified by $| \; |$. In the above analysis of the convergence of an iterative method in a single equation model, this measure was used to indicate the magnitude of the differences between successive iterates. The corresponding measure for a vector or a matrix is called a norm, and is denoted by $|| \; ||$. Norms provide a measure of "distance" in higher dimensions and are useful both for defining convergence and in analyzing under what circumstances convergence will take place.

A vector norm is defined by the following properties, which are analogous to the properties of the absolute value.[6/] Let X and Y be vectors, α a scalar, and $\underline{0}$ a vector all of whose elements are zero. Then a vector norm must satisfy

a. $|| X || > 0$ if $X \neq \underline{0}$; $||X|| = 0$ if $X = \underline{0}$;

b. $||\alpha X|| = |\alpha| \cdot ||X||$;

c. $||X + Y|| \leq ||X|| + ||Y||$.

Matrix norms must also satisfy properties a.-c. (where X,Y and $\underline{0}$ are matrices) as well as

d. $||XY|| \leq ||X|| \cdot ||X||$.

A matrix norm and a vector norm are said to be mutually consistent if, where A is any matrix and X any vector (of suitable dimension)

e. $||AX|| \leq ||A|| \cdot ||X||$.

There are an infinite number of vector and matrix norms that satisfy a.-e. For instance, the familiar Euclidean vector norm,

$$||X|| = \sqrt{(x_1)^2 + (x_2)^2 + \dots + (x_n)^2},$$

where the x_i, i=1, ... , n, are elements of the vector X, is only one member of the so-called L_p family of norms. These norms are defined as

$$||X||_p = (|x_1|^p + |x_2|^p + \dots + |x_n|^p)^{1/p}, \ 1 \leq p \leq \infty.$$

It can now be stated that an iterative method converges to a solution, X^*, of a system of simultaneous equations when, for some vector norm, $||X^* - X^{(k)}||$ approaches zero as k goes to infinity. Viewing X^* as a fixed point of the iterative method, a convergent method can also be defined by the criterion that $||X^{(k+1)} - X^{(k)}||$ approaches zero as k goes to infinity. The use of vector and matrix norms in analyzing (as opposed to merely defining) convergence will be deferred until some other useful concepts have been defined.

2. The Jacobian Matrix.--Convergence of the one-dimensional iterative method $x^{(k+1)} = g(x^{(k)})$ depends on the first derivative of $g(x^{(k)})$ being of absolute value less than one in a neighborhood of the solution point x^*. To generalize this result to the n-equation case requires an extension of the concept "first derivative" to systems of equations in several variables. For one function of n variables, $g(x_1, x_2, \dots, x_n) = 0$, an appropriate generalization is the tangent hyperplane defined by the vector of first partial derivatives of the function, $g'(x_1, x_2, \dots, x_n) =$

$$\left[\frac{\partial g}{\partial x_1}(x_1, x_2, \dots, x_n), \frac{\partial g}{\partial x_2}(x_1, x_2, \dots, x_n), \dots, \frac{\partial g}{\partial x_n}(x_1, x_2, \dots, x_n)\right]$$

For a system of n functions in n variables, the appropriate generalization is the system of tangent planes formed by taking all the first partial derivatives of each function. These first partial derivatives are arranged in a matrix called the Jacobian matrix. It is an n x n matrix whose ij th element is the j th partial derivative, evaluated at some point $\bar{X}$, of the i th equation in a system of n equations each expressed as $g_i(x_1, x_2, \dots, x_n) = 0$.

The extension of this concept to iterative methods is as follows. The iterative method $X^{(k+1)} = G(X^{(k)})$ is composed of n equations in n unknowns and can be expressed in more detail as

$$x_1^{(k+1)} = g_1(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$$
$$x_2^{(k+1)} = g_2(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$$
$$\vdots$$
$$x_n^{(k+1)} = g_n(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$$

The Jacobian whose ij th element is formed by evaluating the j th partial derivative of the i th equation above at the point $(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$ is referred to as the Jacobian of the iterative method at $X^{(k)}$.

3. Spectral Radius.--The spectral radius of an nxn matrix A is denoted by $\rho(A)$ and is equal to the largest of the absolute values of the eigenvalues[7/] of A. That is, $\rho(A) = \max_{1 \leq i \leq n} |\lambda_i(A)|$, where $\lambda_i(A)$ is the i th eigenvalue of A.

4. Contraction Mapping.--The central concept of the general theory of convergence of an iterative method is the contraction mapping. It allows for generalization to higher dimensions of the rule that a one-dimensional iterative method will converge if $x^{(0)}$ lies in a neighborhood of the solution, x^*, where $|g'(x^{(k)})| < 1$. Before stating the theorems that contain this result, contraction mappings will be explained.

A mapping refers to a function G whose domain and range are subsets of n-dimensional real space, R^n.[8/] The function assigns to each n-coordinate point in the domain an n-coordinate point from the range. The function is thus said to "map" the domain into the range.

Let the domain of a mapping G be a subset of R^n called D. The mapping

G is said to be contractive on a subset of its domain, $D_0 \subset D$, if there is a scalar $\alpha < 1$ such that $||G(Y) - G(Z)|| < \alpha \; ||Y - Z||$ for all points Y and Z in D_0 /̲17, p. 120̲/. This means that the mapping "shrinks" the domain down into a range which is "smaller" than the domain.

Consider an iterative function as a mapping. Since $X^{(k)} = G(X^{(k-1)})$ and $X^{(k+1)} = G(X^{(k)})$, it can be seen that the range of one iteration is the domain of the next. This fact can have important implications for a contractive iterative method. For instance, if an iterative function G is contractive over $D_0 \subset D$, then there exists an $\alpha < 1$ such that

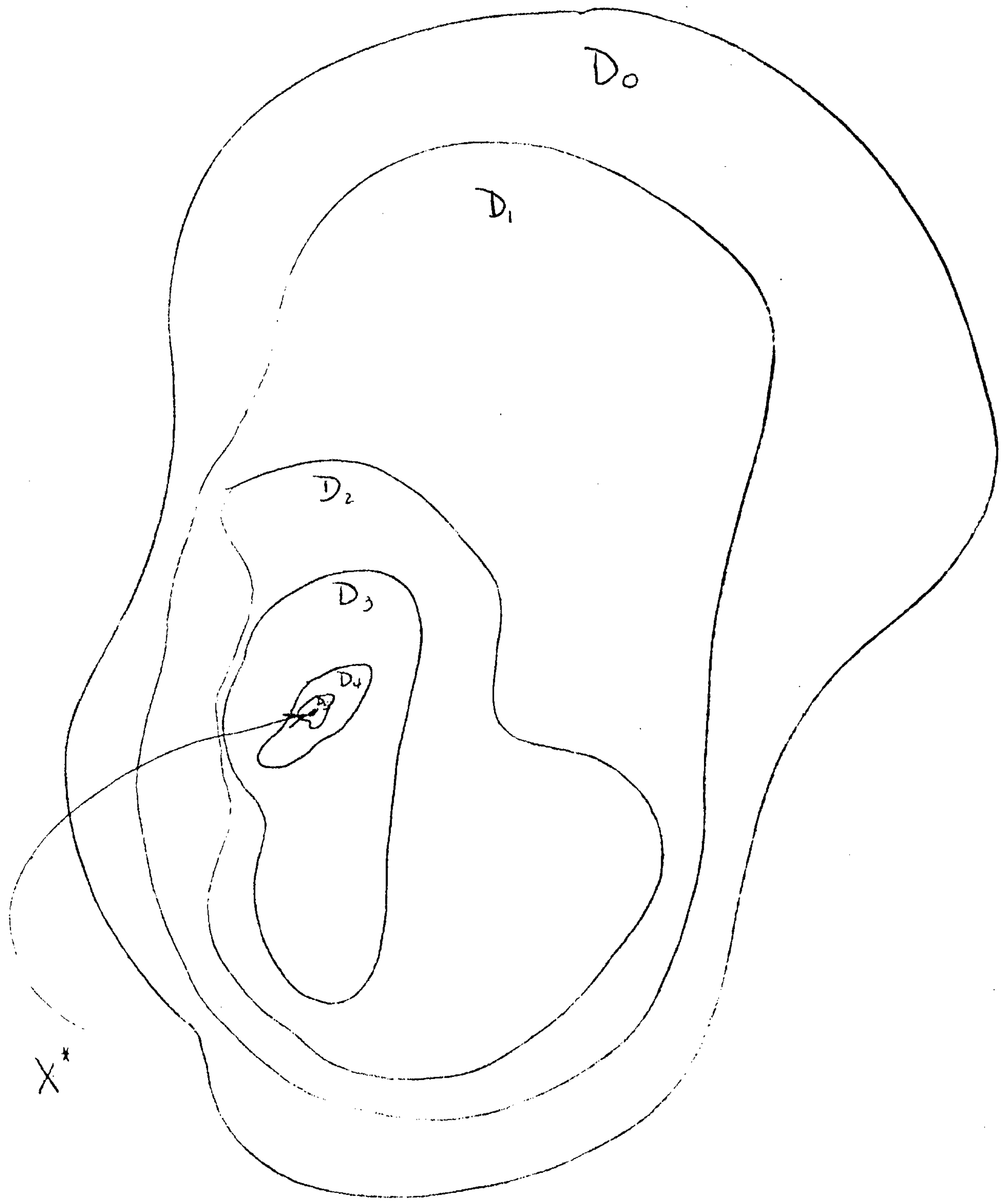
$$||X^{(k+1)} - X^{(k)}|| = ||G(X^{(k)}) - G(X^{(k-1)})|| < \alpha \; ||X^{(k)} - X^{(k-1)}||$$

for all $X^{(k-1)}$ and $X^{(k)}$ in D_0. That is, a contractive iterative function reduces the successive first differences in the sequence of iterates. If $X^{(k+1)}$ also belongs to D_0 (i.e., G maps D_0 into itself), then this process will continue and the successive differences will go to zero. This is the reasoning behind the following contraction mapping theorem /̲17, pp. 119-121, 385̲/: If G, defined over domain D, maps R^n into R^n and is contractive on a closed set $D_0 \subset D$, and $G(D_0) \subset D_0$, the sequence of iterates $X^{(k)}$ corresponding to any initial point $X^{(0)}$ in D_0 is well-defined and converges to a unique fixed point X^* in D_0. Note that global convergence occurs when $D_0 = D$.

Figure 3 represents a mapping contractive over the subset D_0. D_1 is the range of D_0 and lies within and is smaller than D_0. This is true of all successive domains and ranges which are gradually constricted to smaller and smaller regions around the fixed solution point X^*.

5. Convergence Theorems.--Since the property of being contractive is so important to the convergence of an iterative method, it is also important to know when an iterative method is contractive or how it can be made so.

Figure 3



The following two theorems show that certain characteristics of the Jacobian of an iterative method are sufficient to make it contractive.

(1) If a mapping G of R^n has a Jacobian, $G'(X)$, such that for some norm $||G'(X)|| \leq \alpha < 1$ for all X in a convex subset of the domain of G, then G is contractive on that subset.[9/]

(2) If the spectral radius of an n x n matrix A is less than one, then there is a norm of A which is also less than one. That is, if $\rho(A) < 1$, then $||A|| < 1$ for some norm.[10/]

D. Summary

These two convergence theorems imply that an iterative method is contractive over some subset of its domain if the spectral radius or any norm of the Jacobian of the iterative method is less than one over that subset.[11/] While contractiveness does not insure convergence (the iterative method must also map D_0 into itself), it would be advantageous to know how to make a particular iterative method contractive when applied to a particular model. However, the results of the theorems above are difficult to apply except in simple cases such as small linear systems. In general, users of iterative methods rely on "rules-of-thumb" such as those reported in Parts II and III. Only some of these practical rules of thumb can be mathematically related to the theorems of convergence, but keep in mind that all modifications are aimed at making the iterative method into a contraction mapping.

Part II: The Jacobi and Gauss-Seidel Methods

A. Explanation of the Methods

Use of the Jacobi and Gauss-Seidel methods was, for many years, limited to linear equation systems. In recent years they have been slightly modified and successfully applied to nonlinear systems as well [12; 17, p. 222]. The exposition here will parallel the historic development, defining and analyzing these methods for linear systems and generalizing to nonlinear problems.

Both the Jacobi and Gauss-Seidel methods are very straightforward applications of a trial-and-error strategy. Consider the linear equation system AX = B, where A is an n x n matrix of constants, X is an n x 1 vector of unknowns, and B is an n x 1 vector of constants. Exact methods such as matrix inversion or Gaussian elimination could be used to calculate a solution. However, when A is large and contains many zeros, exact methods may be computationally inferior to iterative methods.12/

The first step in the Jacobi method for solving the linear equation system AX = B is to express each x_i, i = 1, ... , n, as a function of the other x's. Rearranging the ith equation,

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{ii}x_i + \dots + a_{in}x_n = b_i, \text{ yields}^{13/}$$

$$x_i = \left[b_i - \sum_{j \neq 1} a_{ij}x_j \right] / a_{ii}$$

These equations are used to generate new iterates from previous iterates by substituting into the right-hand sides values of $x_j^{(k)}$, j=1, ... , i-1, i+1, ... , n. The left-hand sides become the elements of the new iterates, or

$$x_1^{(k+1)} = (b_i - \sum_{j\neq 1} a_{1j}x_j^{(k)})/a_{11}$$

$$x_2^{(k+1)} = (b_2 - \sum_{j\neq 2} a_{2j}x_j^{(k)})/a_{22}$$

$$\vdots$$

$$x_n^{(k+1)} = (b_n - \sum_{j\neq n} a_{nj}x_j^{(k)})/a_{nn}.$$

For instance, the first iterate, $X^{(1)}$, is generated by substituting the elements of the initial guess, $X^{(0)}$, for the variables on the right-hand side of the equations above. The values for the second iterate, $X^{(2)}$, are obtained by substituting the just-calculated elements of $X^{(1)}$ into the right-hand side of the equations. The continuation of this process yields a sequence of Jacobi iterates, $\{X^{(k)}\}$, $k = 0, 1, 2, \ldots$.

The Jacobi method is "conservative" in its incorporation of new "information". In iteration k + 1, each element of $X^{(k+1)}$ is calculated independently of the other elements of $X^{(k+1)}$. Each is a function only of the previous iterate. For this reason $X^{(k+1)}$ is unaffected by the order in which its elements, $x_i^{(k+1)}$, are calculated.

However, suppose the sequence of iterates is approaching X^*. Then $x_i^{(k+1)}$ is closer to x_i^* than $x_i^{(k)}$, and it could be used to improve the calculations of the remaining elements, $x_j^{(k+1)}$, $i < j \leq n$, of the new iterate. For example, suppose that the first element of the new iterate, $x_1^{(k+1)}$, has already been calculated as in the Jacobi method above. Then in the calculation of $x_2^{(k+1)}$ it would be possible to substitute $x_1^{(k+1)}$ for the $x_1^{(k)}$ which is used in the Jacobi method. If the sequence of iterates were converging and $x_1^{(k+1)}$ were closer to x_1^* than $x_1^{(k)}$, this substitution might result in a value closer to x_2^* than the value that would be calculated using the Jacobi method. Similarly, $x_1^{(k+1)}$ and $x_2^{(k+1)}$ would be used to calculate $x_3^{(k+1)}$; $x_1^{(k+1)}$, $x_2^{(k+1)}$, and $x_3^{(k+1)}$ would be used to calculate $x_4^{(k+1)}$, and so on. This method, which replaces $x_i^{(k)}$ in the Jacobi method by $x_i^{(k+1)}$ as soon as each new element of the new iterate becomes available, is known as the Gauss-Seidel method. For the linear equation system AX = B, the Gauss-Seidel method is algebraically expressed as

$$x_1^{(k+1)} = (b_1 - \Sigma_{j\geq 2}\, a_{1j}x_j^{(k)})/a_{11}$$

$$x_2^{(k+1)} = (b_2 - a_{21}x_1^{(k+1)} - \Sigma_{j\geq 3}\, a_{2j}x_j^{(k)})/a_{22}$$

$$\vdots$$

$$x_i^{(k+1)} = (b_i - \Sigma_{j\leq (i-1)} a_{ij}x_j^{(k+1)} - \Sigma_{j\geq (i+1)}\, a_{ij}x_j^{(k)})/a_{ii}$$

$$\vdots$$

$$x_n^{(k+1)} = (b_n - \Sigma_{j\leq (n-1)} a_{nj}x_j^{(k+1)})/a_{nn}$$

As a result of this rapid incorporation of "new information" the Gauss-Seidel method may converge in fewer iterations than the Jacobi method. However, if some element of $X^{(k+1)}$ is a poor estimate of the corresponding element of the solution vector, this error will be incorporated in the subsequent elements of $X^{(k+1)}$. For this reason, the Gauss-Seidel method is more sensitive to a poor starting guess and may fail to converge in a problem that could be solved using the Jacobi method. Furthermore, the order of the equations, which is of no importance in the Jacobi method, may strongly affect the Gauss-Seidel method. Some orders may produce convergence, others may not. Strategies for ordering the equations in a Gauss-Seidel method are given in Section D.

The Jacobi and Gauss-Seidel methods are easily extended to nonlinear systems of the form $F(X) = \underline{0}$, where X is an n x 1 vector of variables, and F denotes the system of n equations in n unknowns, $f_i(x_1, x_2, \ldots, x_n) = 0$. The only modifications required are due to the fact that in transforming the system $F(X) = \underline{0}$ into the form $X^{(k+1)} = G(X^{(k)})$ of the iterative methods, the equations

$$x_i^{(k+1)} = g_i(x_1^{(k)}, \ldots, x_{i-1}^{(k)}, x_{i+1}^{(k)}, \ldots, x_n^{(k)})$$

of the Jacobi method, and the equations

$$x_i^{(k+1)} = g_i(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, x_{i+1}^{(k)}, \ldots, x_n^{(k)})$$

of the Gauss-Seidel method become nonlinear. It is thus impossible to state a general formula for transforming the system $F(X) = \underline{0}$ to the form, $X^{(k+1)} = G(X^{(k)})$, that is required for iteration. In this case the user himself must supply an algebraic statement of the model with each endogenous variable appearing exactly once on the left-hand side of an equation.[14/]

Transforming the system to the Jacobi or Gauss-Seidel iterative form requires more of the user than algebraic skill, however, In both the linear and nonlinear cases these methods are very sensitive to the pattern of <u>normalization</u> chosen by the user. Normalization refers to the rewriting of an equation so that one of its endogenous variables is isolated on the left-hand side of the equal sign. The pattern of normalization of a system of equations refers to the choice of which equation to use to solve for each endogenous variable so that each endogenous variable appears exactly once on the left-hand side of an equation. (Users are cautioned that equations must be normalized independently. No expressions for any variable in the equation can be substituted in from other equations!) According to the previous discussion, in linear equation systems the pattern of normalization is specified by the rule of solving the ith equation for the ith variable. The user controls the pattern of normalization by his choice of the ordering of the equations, 1, 2, ... , n, that appear in the system AX = B. In the nonlinear case the user's control is even more apparent since he must also perform the inversion of the equations himself. In both cases, however, the iterative methods may converge for some patterns of normalization and diverge for others. For instance, in solving the model $F(X) = \underline{0}$, the normalization

$$\begin{aligned}
x_1 &= f_1^{-1}\ (x_2, x_3, \ldots , x_n) \equiv g_1(x_2, x_3, \ldots , x_n)\\
x_2 &= f_2^{-1}\ (x_1, x_3, \ldots , x_n) \equiv g_2(x_1, x_3, \ldots , x_n)\\
&\vdots\\
x_n &= f_n^{-1}\ (x_1, x_2, \ldots , x_{n-1}) \equiv g_n(x_1, x_2, \ldots , x_{n-1}),
\end{aligned}$$

where f_i^{-1} denotes the result of normalizing the ith equation of the system $F(X) = \underline{0}$ and g_i refers to the ith function component of the iterative method $X^{(k+1)} = G(X^{(k)})$, may converge. At the same time the alternative normalization

$$\begin{aligned}
x_1 &= f_n^{-1}\ (x_2, x_3, \ldots , x_n) \equiv gg_1(x_2, x_3, \ldots , x_n)\\
x_2 &= f_{n-1}^{-1}\ (x_1, x_3, \ldots , x_n) \equiv gg_2(x_1, x_3, \ldots , x_n)\\
&\vdots\\
x_n &= f_1^{-1}\ (x_1, x_2, \ldots , x_{n-1}) \equiv gg_n(x_1, x_2, \ldots , x_{n-1})
\end{aligned}$$

where the gg_i denotes the iterative function components of the alternative iterative method $X^{(k+1)} = GG(X^{(k)})$, may also converge. On the other hand, one or both of these normalizations may lead to divergence. A normalization which leads to convergence of the Jacobi or Gauss-Seidel method may be difficult to determine. The problem of choosing an appropriate pattern of normalization is illustrated in the examples in Section C.

B. Convergence and Normalization in a Simple Model.

Figure 4 depicts the following model of price and quantity marketed of a commodity:

Supply $q = 2 + 2p$

Demand $q = 12 - 3p$

There are two equations in two endogenous variables, quantity (q) and price (p). Simple algebra reveals that the solution is q = 6, p = 2, represented by point A in Figure 4.

To discover this solution by Gauss-Seidel iteration, one of the equations must first be rewritten to express p as a function of q. In other words, one of the equations must be normalized on p. There are two possible patterns of normalization, one with the demand equation normalized on p and the supply equation normalized on q and the other vice versa. After labeling the variables with time superscripts to distinguish new iterates from old iterates, the two patterns[15/] can be written as follows:

1. (The Clockwise Pattern or "Cobweb Model")

   Supply $q^{(t)} = 2 + 2p^{(t-1)}$

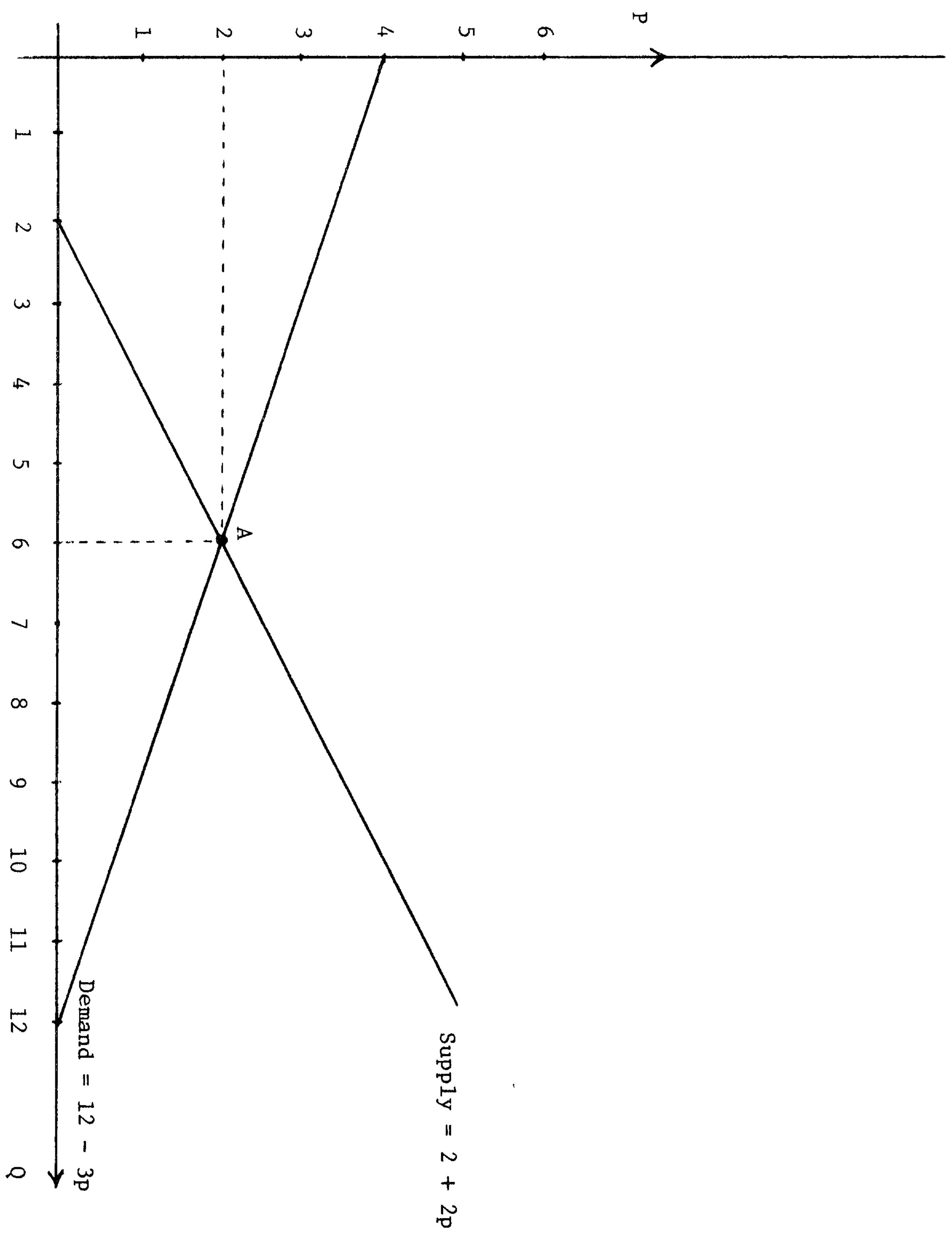
   Demand $p^{(t)} = (\frac{-1}{3})q^{(t)} + 4$

2. (The Counter-clockwise Pattern)

   Supply $p^{(t)} = (\frac{1}{2})q^{(t-1)} - 1$

   Demand $q^{(t)} = 12 - 3p^{(t)}$.

Figure 4



Pattern 1, as indicated, corresponds to what agricultural economists call the cobweb model. Certain causal links between supply and demand are implicit in the pattern of normalization of the cobweb model. The supply function determines the quantity produced (based on a predetermined price) and the demand function determines the price at which the commodity is sold (based on a predetermined quantity). Graphically, the Gauss-Seidel method in the cobweb model moves <u>vertically from S to D</u> and <u>horizontally from D to S</u>, resulting in the clockwise cobweb shown in Figure 5. Note that in this problem the cobweb pattern of normalization yields a sequence of price-quantity guesses which tend to converge to the solution at point A.

In pattern 2, the supply function determines price (based on predetermined quantity) and the demand function determines quantity (based on predetermined price). Progression is <u>horizontal from S to D</u> and <u>vertical from D to S</u>, resulting in the counterclockwise movement and divergent price quantity sequence shown in Figure 6. Gauss-Seidel iteration would solve the model under the first pattern of normalization but not under the second. Another analogy from economics would be a model with one equilibrium solution and two alternative adjustment processes, one which is stabilizing and one which is destabilizing.

Normalization in the linear two-equation case is examined in more mathematical detail in Appendix B. For more equations or nonlinear models it is increasingly difficult to precisely determine convergent normalizations, and sensitivity to normalization is a serious drawback of the Gauss-Seidel and Jacobi methods. Suggestions on how to normalize models and how to reduce the sensitivity of the methods to the pattern of normalization are discussed in the next section.

FIGURE 5

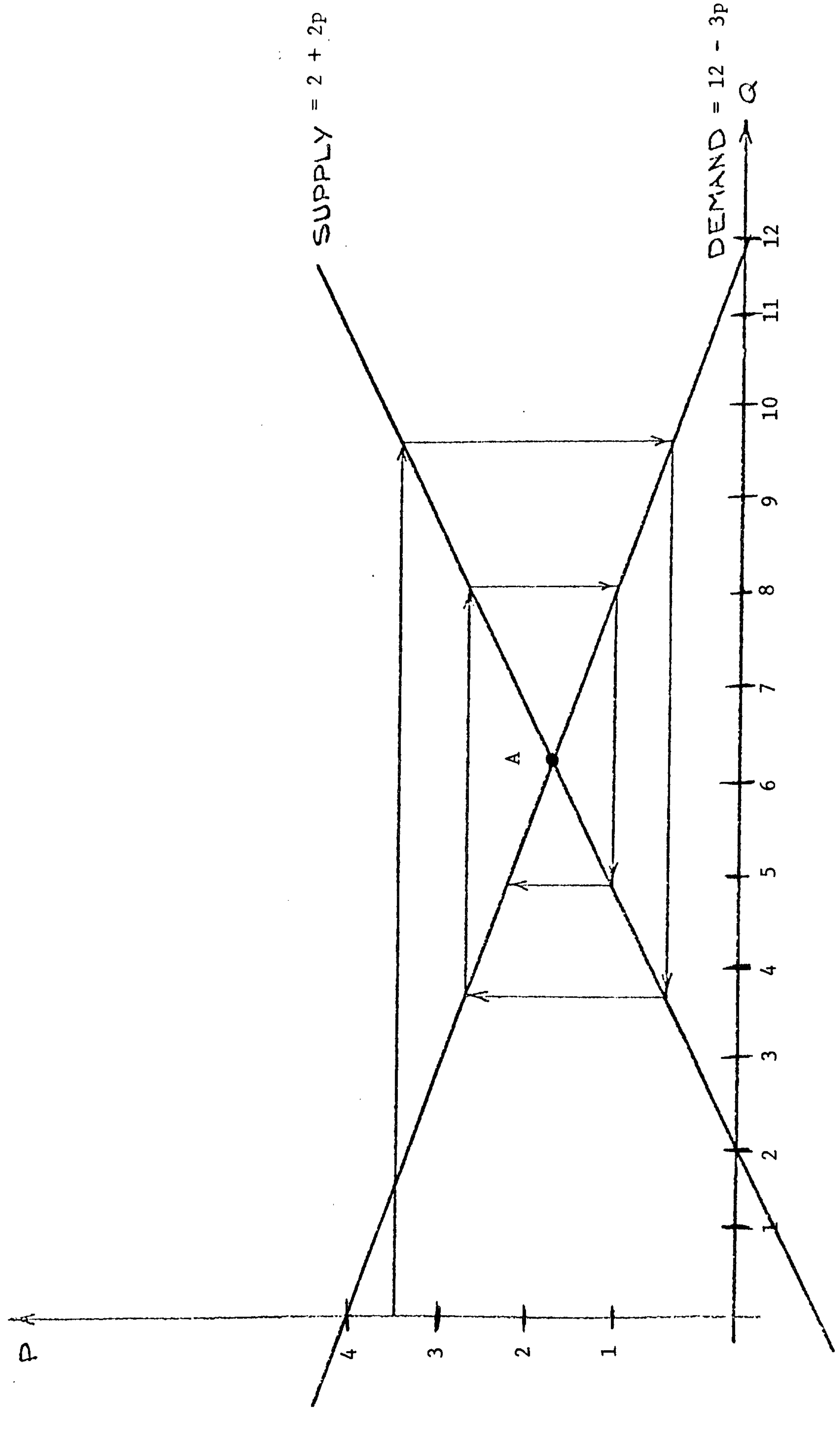
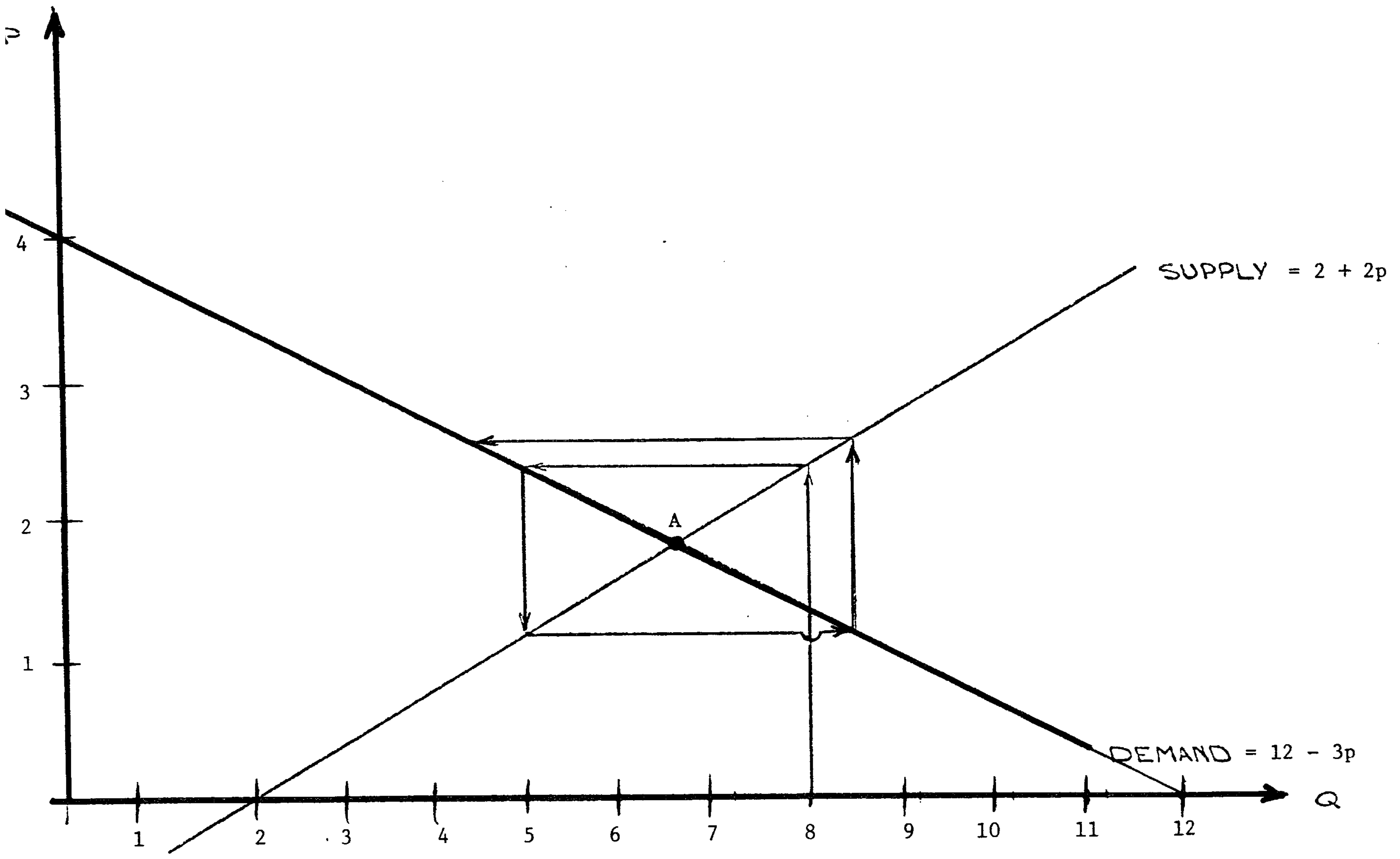


FIGURE 6



C. Implementing the Jacobi and Gauss-Seidel Methods

Jacobi and Gauss-Seidel methods are mathematically similar, and much of the same advice applies to both of them. Most of the suggestions below are derived from the experience of economists in USDA's Economic Research Service, who have been using Jacobi and Gauss-Seidel methods to solve agricultural subsector models with up to 100 linear and nonlinear simultaneous equations. A summary of their conclusions appears in [12].

In work on similar agricultural subsector models (feed grains in particular) at the Department of Agricultural and Applied Economics, University of Minnesota, researchers have observed a significant advantage of the Jacobi method over the Gauss-Seidel in achieving convergence. While most of the following suggestions apply to either method, it is thus recommended that when the Gauss-Seidel method diverges the researcher should try the Jacobi method before reordering or renormalizing the equations in hopes of obtaining a Gauss-Seidel solution. Only where the same model is to be solved a great many times may the slightly more rapid rate of convergence of the Gauss-Seidel method justify the effort needed to reformulate the model to obtain Gauss-Seidel convergence.

1. Picking an initial guess.--The user of any iterative method must use whatever evidence is available to help select an accurate initial guess. If the model to be solved is meant to simulate an actual economy for which historical data are available on the endogenous variables, the values of those variables that prevailed in a relevant historical time period frequently constitute a good initial guess. However derived, an initial guess can often be improved by insuring that the values chosen satisfy all linear identities (e.g., resource constraints) in the model.

2. Program debugging.--An advantage of the Jacobi method is that it can be used to check for errors in the computer program. This is accomplished by stopping the method after one iteration and comparing the first iterate calculated by the computer with the first iterate expected by the user. Deviations indicate a programming error. In models that have been estimated using regression analysis and for which the initial guess is set equal to the observed values for a given time period, the residual between the first iterate and the initial guess must be identical to the residual in the regression on which the equation is based.

3. Ordering of the equations.--As pointed out previously, the order of equations does not affect the Jacobi method but greatly affects the Gauss-Seidel. Heien, Mathews, and Womack /12, p. 73/ suggest ordering the equations as recursively as possible. This can be accomplished by writing the relationship between the endogenous variables in matrix form, using the number 1 to indicate that a variable appears in a certain equation and the number 0 to indicate that it does not. The example presented in their article would be written as in Figure 7. The first row, for instance, indicates that the first equation contains x_4 and x_1 and that it has been normalized with x_4 appearing on the left as a function of x_1.

Figure 7. Interdependence Array

	Variables				
Equation	x_1	x_2	x_3	x_4	x_5
x_4	1			1	
x_5		1			1
x_3	1	1	1		
x_2		1	1		
x_1	1	1			1

The suggested strategy is to reorder the rows until the most nearly lower triangular matrix is found. The order which yields the matrix with fewest elements above the main diagonal is the first order to try when implementing the Gauss-Seidel method.

This procedure can be tedious for large models or models that are frequently modified. Dutch macroeconomists have developed a similar procedure which has the advantage of being easily carried out on a computer. A computer program for performing this ordering scheme appears in Appendix C.

If the first ordering does not converge, other nearly triangular orderings may be tried. If convergence is still not achieved, it may be necessary to segregate the full model into several smaller systems. Further attention, including renormalization of some equations, can then be given to those sub-systems that continue to diverge. Variables may also be "sequentially exogenized" (replaced, one at a time, by constants) to find the equations which are causing divergence.

4. Normalization.--One requirement of both the Jacobi and Gauss-Seidel method is that the model be written so that each endogenous variable appears on the left side of the equality sign exactly once. This can be doen in as many as n! ways for an n x n system, and often there are few *a priori* criteria for selecting a normalization that will yield convergence under the Jacobi or Gauss-Seidel method. Techniques for finding a convergent normalization range from the very sophisticated to trial-and-error methods such as sequentially exogenizing variables or segregating the model into submodels and confining attention to only those submodels that diverge. In the case of a model that has been econometrically estimated, Heien, Mathews, and Womack recommend that the dependent variable in each equation should be the

same as the variable that was dependent when the parameters were estimated. Their recommendation may be generalized by urging that the pattern of normalization reflect, as much as possible, the pattern of causality believed to prevail in the market or markets being studied. USDA experience, however, indicates that the problem of normalization in the simple Jacobi or Gauss-Seidel method can often be circumvented by the introduction of a dampening factor.

5. Dampening factors.--The use of dampening factors makes the new iterate a weighted average of the previous iterate and the iterate that would have been chosen by the undampened method. In effect, the direction of movement is the same as in the undampened method, but the dampening factor controls how far along that direction the new iterate should lie. When the dampening factor is less than one, this conservative use of new information "helps prevent a divergent arrangement of equations from dominating the system. A dampening factor in effect allows other equations more rounds to converge and tends to pull the divergent arrangement back toward convergence." [12, p. 74].

Dampened versions of both the Gauss-Seidel and Jacobi methods are easily constructed. The dampened version of the Gauss-Seidel is frequently referred to as the successive overrelaxation method (SOR) [17, p. 215], and the dampening factor in this case is called the relaxation parameter. Letting subscripts SOR and GS denote SOR and Gauss-Seidel iterates respectively, the SOR method can be written $X_{SOR}^{(k+1)} = (1 - w)\ X^{(k)} + (w)\ X_{GS}^{(k+1)}$, where w is the dampening or relaxation parameter. If w = 1, the SOR method becomes identical to the Gauss-Seidel. Similarly, the dampened version of the Jacobi can be written $X_{DJ}^{(k+1)} = (1 - w)\ X^{(k)} + (w)\ X_{J}^{(k+1)}$, where subscripts DJ and J stand for the dampened Jacobi and Jacobi methods, respectively.

The effect of a dampening factor on an iterative method is to alter the rate of convergence. Despite their name, dampening factors do not always slow down the rate of convergence. They may speed it up or even achieve convergence in cases where the undampened method diverged. It can be shown that in linear systems only dampening factors between 0 and 2 can produce convergence of the Gauss-Seidel method, and in certain restricted cases it is possible to calculate the exact dampening factor that would maximize the rate of convergence [4, p. 193].

In most cases, no attempt is made to select *a priori* an optimal dampening factor. Researchers in USDA and at the Department of Agricultural and Applied Economics of the University of Minnesota have used the Jacobi method with a dampening factor of 0.25 applied to each equation. This puts 75 percent of the weight in each iteration on the previous iterate, but results in a method of great stability and moderate cost. Experience with the method has been limited to fairly small (15 to 100 equations) and highly linear models, but convergence has been generally obtained without reordering or renormalizing the equations.

6. *Variable dampening factors*.--While USDA researchers have been successful using the same dampening factor (0.25) for each equation and each iteration, the dampening factor may be varied with each equation or iteration. Selecting a different but constant dampening factor for each equation is simple to program and has the advantage of allowing special treatment of segments of the model thought to be causing divergence. A

simple example of such a strategy, which illustrates some of the properties of all dampened iterative methods, can again be taken from agricultural economics. The combination of the cobweb model with the partial adjustment of supply hypothesis gives a price determination model exactly equivalent to a Gauss-Seidel method with one dampened equation. Consider this modification of the previous model of supply and demand:

demand $q^{(t)} = 8 - 2p^{(t)}$, or $p^{(t)} = \frac{q^{(t)}-8}{2}$

desired supply $q^{(t)*} = -4 + 4p^{(t-1)}$

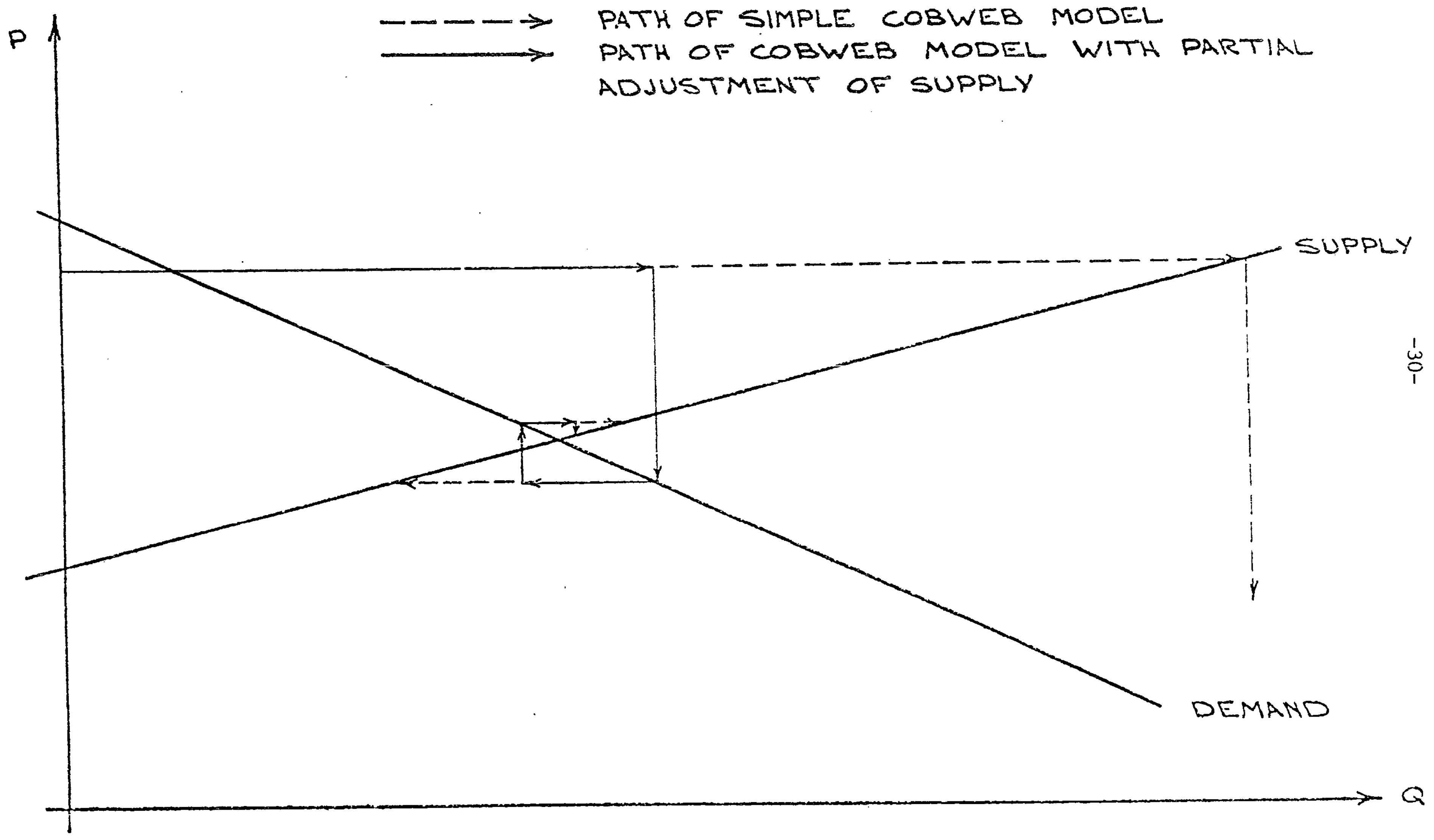
supply actually achieved given limitations on production adjustments $q^{(t)} = q^{(t-1)} + \delta(q^{(t)*} - q^{(t-1)}) = (1-\delta)q^{(t-1)} + \delta q^{(t)*}$

As Figure 8 shows, the undampened Gauss-Seidel method, which is equivalent to the simple cobweb model, would diverge. The introduction of a dampening factor of 0.5 on the supply equation, which is equivalent to a cobweb model with partial adjustment of supply, yields a convergent method.

Summary.--Jacobi and Gauss-Seidel methods are simple to program and require relatively little computer storage space. With the addition of dampening factors the Jacobi method especially has proven to converge quite regularly and at moderate cost. The Jacobi method also facilitates program and model debugging. The simplicity and dependability of these methods argue for their continued popularity with economists.

This completes the discussion of two methods originally developed to solve linear systems and subsequently successfully applied to nonlinear systems as well. In the next section two methods intended primarily for nonlinear systems are introduced.

FIGURE 8


PATH OF SIMPLE COBWEB MODEL
PATH OF COBWEB MODEL WITH PARTIAL ADJUSTMENT OF SUPPLY
P
Q
SUPPLY
DEMAND

Part III: Newton-based Methods

A. Explanation of the Methods

The Jacobi and Gauss-Seidel methods are among the least complicated iterative methods. In both, the iterative function $X^{(k+1)} = G(X^{(k)})$ is derived directly from a rearrangement of the equations of the system $F(X) = \underline{0}$.

The methods discussed in this section are more sophisticated in that two steps are employed. First, the system of equations $F(X) = \underline{0}$ is replaced by a system of linear approximating functions. Then the linear system is transformed to the form $X^{(k)} = G(X^{(k)})$.

The first method discussed, approximation by parallel chords, is used more commonly in pedagogy than research because it is the simplest example of a linear approximation technique. Once the general concepts of linear approximation techniques have been introduced with reference to the parallel chord method, discussion will pass to the more complex but also more useful methods of Newton and Brown.

1. Parallel Chord Methods.--Consider the problem of finding a zero for a nonlinear function such as the one depicted in Figure 9. The sequence of iterates $x^{(1)}, \ldots , x^{(5)}$ shown in the graph was generated by a parallel chord method. The basic idea of a parallel chord method for finding the zeros of an implicit function is to replace the function, $f(x)$, by a linear approximation, $\ell(X)$, that passes through the current function value , $f(x^{(k)})$. The exact zero of the linear approximating function (i.e., the value of x that satisfies $\ell(x) = 0$) becomes the new iterate, $x^{(k+1)}$. The linear approximating function can be written as $\ell^{(k)}(x) = a(x - x^{(k)}) + f(x^{(k)})$, for some $a \neq 0$. Setting $\ell^{(k)}(x) = 0$ and solving for $x^{(k+1)} \equiv x$ gives $x^{(k+1)} = x^{(k)} - (a^{-1})\, f(x^{(k)})$.

For n x n systems of nonlinear implicit functions, $F(X) = \underline{0}$, the parallel

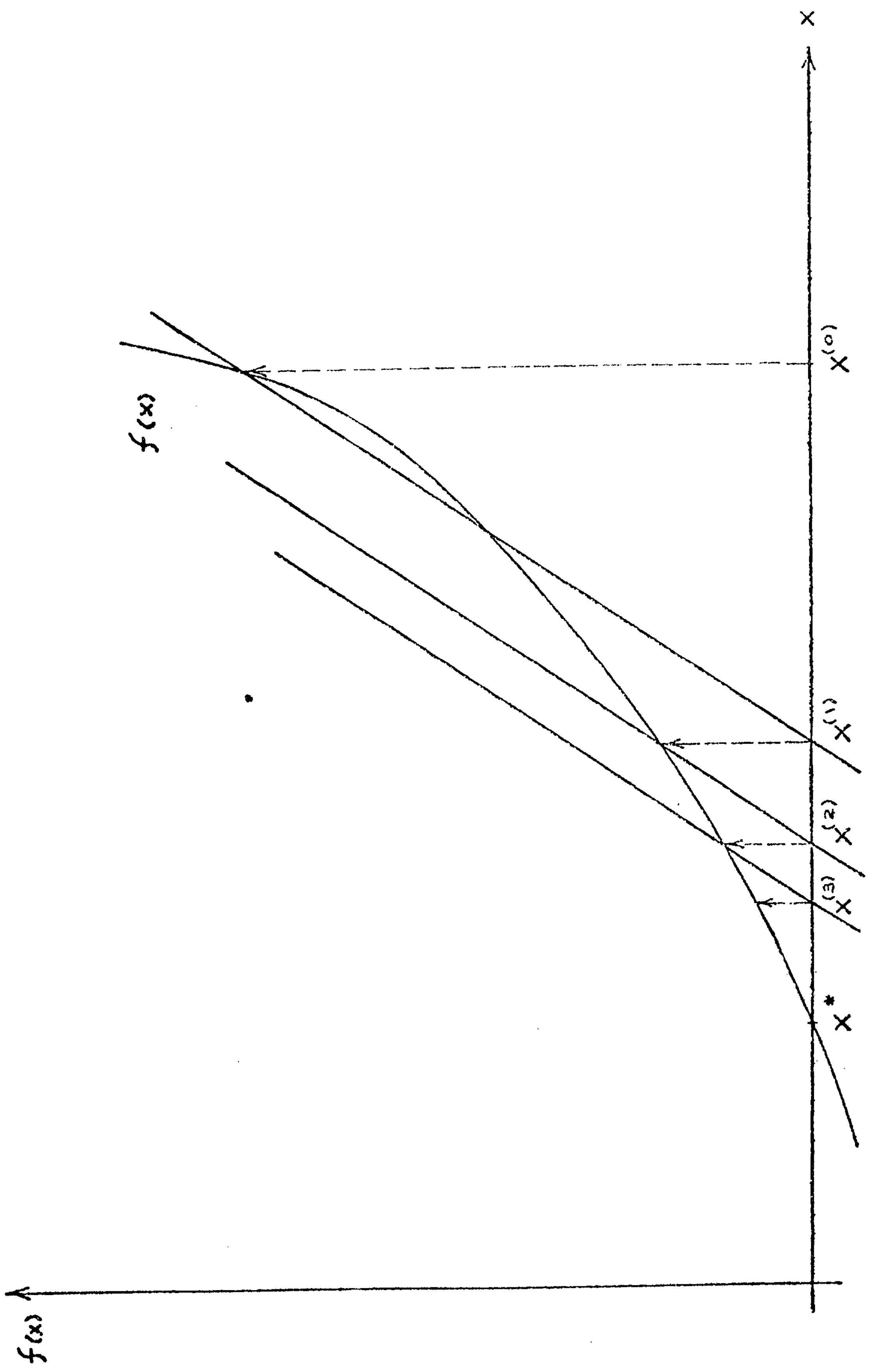


FIGURE 9

chord method is generalized by taking linear approximations (hyperplanes that pass through F(X)) of all n functions and calculating the zeros of the resulting system of linear equations (intersection of the n approximating planes and the plane $F(X) = \underline{0}$). These zeros become the new iterate. The system of linear approximating functions can be written as

$$\ell_1^{(k)}(x_1,\ldots,x_n) = a_{11}(x_1-x_1^{(k)}) + a_{12}(x_2-x_2^{(k)}) + \ldots + a_{1n}(x_n-x_n^{(k)}) + f_1(x_1^{(k)},\ldots x$$

$$\ell_2^{(k)}(x_1,\ldots,x_n) = a_{21}(x_1-x_1^{(k)}) + a_{22}(x_2-x_2^{(k)}) + \ldots + a_{2n}(x_n-x_n^{(k)}) + f_2(x_1^{(k)},\ldots x$$

$$\vdots \qquad\qquad \vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots$$

$$\ell_n^{(k)}(x_1,\ldots,x_n) = a_{n1}(x_1-x_1^{(k)}) + a_{n2}(x_2-x_2^{(k)}) + \ldots + a_{nn}(x_n-x_n^{(k)}) + f_n(x_1^{(k)},\ldots x$$

In matrix notation this becomes

$$L^{(k)}\ (X) = A(X-X^{(k)}) + F(X^{(k)}),$$

where A is the n x n matrix (a_{ij}), and X and $X^{(k)}$ are the n x 1 vectors (x_i) and $(x_i^{(k)})$, respectively. Solving $L^{(k)}\ (X) = 0$ for $X \equiv X^{(k+1)}$ gives $X^{(k+1)} = X^{(k)} - A^{-1}\ F(X^{(k)})$.

Linear but non-parallel chord methods result when the matrix A is allowed to vary with each iteration. Then $L^{(k)}(X) = A^{(k)}(X - X^{(k)}) + F(X^{(k)})$. Many such linearization methods have been devised and implemented by numerical analysts. They differ from one another in their choice of linear approximating functions and in the methods used to calculate the zeros of the resulting system of linear equations. The two linearization methods discussed below are Newton's method and Brown's method. The Newton, or Newton-Raphson, method is a well-known and widely used technique. It has been modified in many ways to produce a variety of "Newton-based" algorithms. One of these modifications is Brown's method, which substantially reduces computer storage requirements and has been shown to be quite reliable in solving economic models. Familiarity with either of these methods will introduce the applied economist to a large class of valuable iterative methods.[16/]

2. Newton's and Brown's Methods.--In Newton's method the matrix $A^{(k)}$ is updated for each iteration. The linear approximations are the tangents to the functions at the point $F(X^{(k)})$. Consider the sequence of Newton iterates for a one-dimensional problem as shown in Figure 10. The value $x^{(0)}$ is supplied by the user and $f(x^{(0)})$ is calculated. Then the equation for the line tangent to f(x) at $f(x^{(0)})$ is written,

$$y = f(x^{(0)}) + (x-x^{(0)}) \cdot f'(x^{(0)}).$$

The value of x that sets y = 0 is

$$x = x^{(0)} - \frac{1}{f'(x^{(0)})} \cdot f(x^{(0)}).$$

This value of x becomes $x^{(1)}$, and the process begins again. Thus, the one-dimensional case is

$$x^{(k+1)} = x^{(k)} - \frac{1}{f'(x^{(k)})} \cdot f(x^{(k)}).$$

The one-dimensional case is useful for illustrating some of the problems of the Newton method and related techniques like Brown's method. Figure 11 below shows the case where $f'(x^{(k)}) = 0$. In this situation the tangent line does not intersect the x-axis and the Newton method is undefined. Figure 11b shows that the tangent at some $x^{(k)}$ may be a very poor approximation of f(x), leading to a poor selection of the new iterate $x^{(k+1)}$ and possibly causing

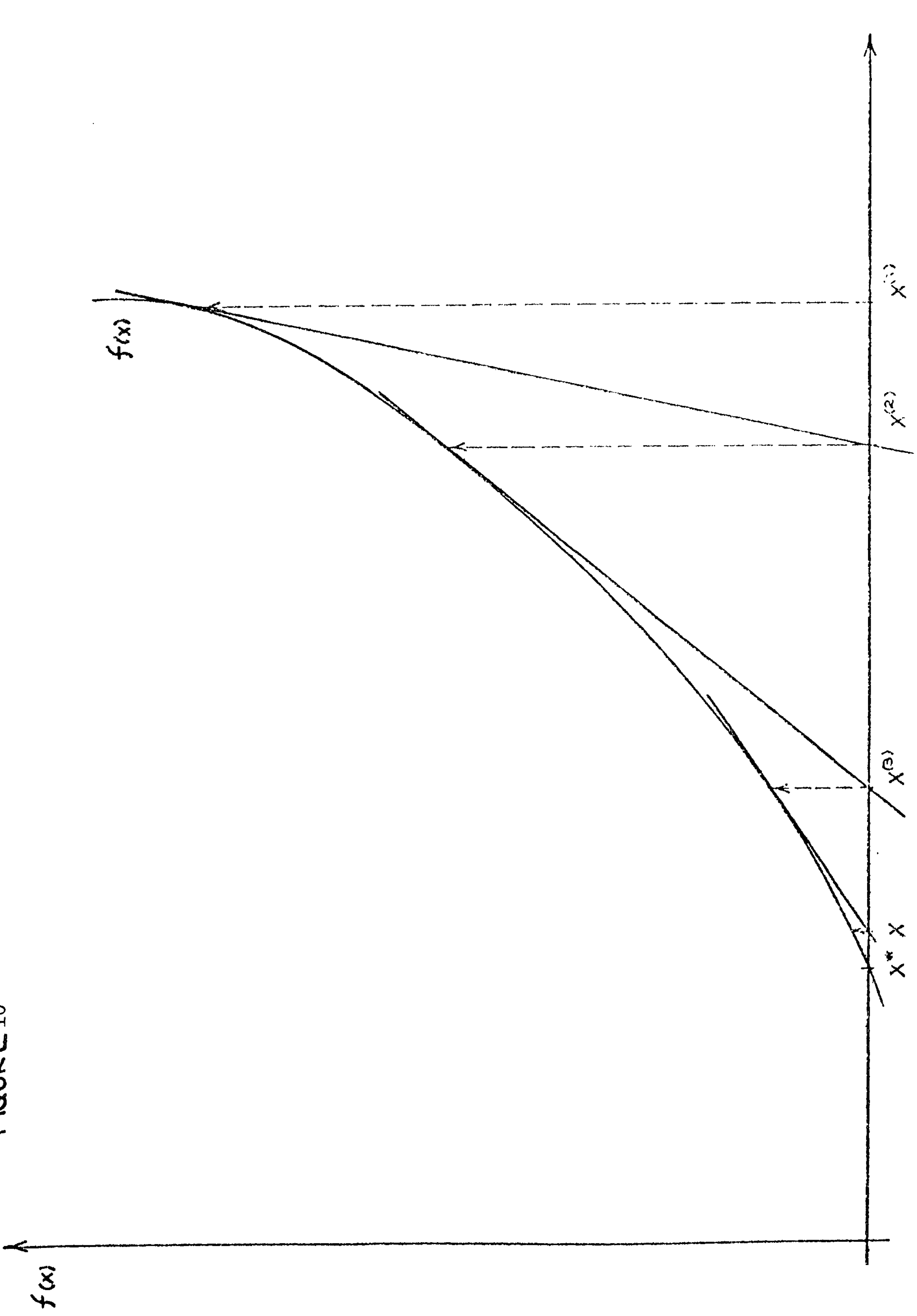


FIGURE 10

FIGURE 11A

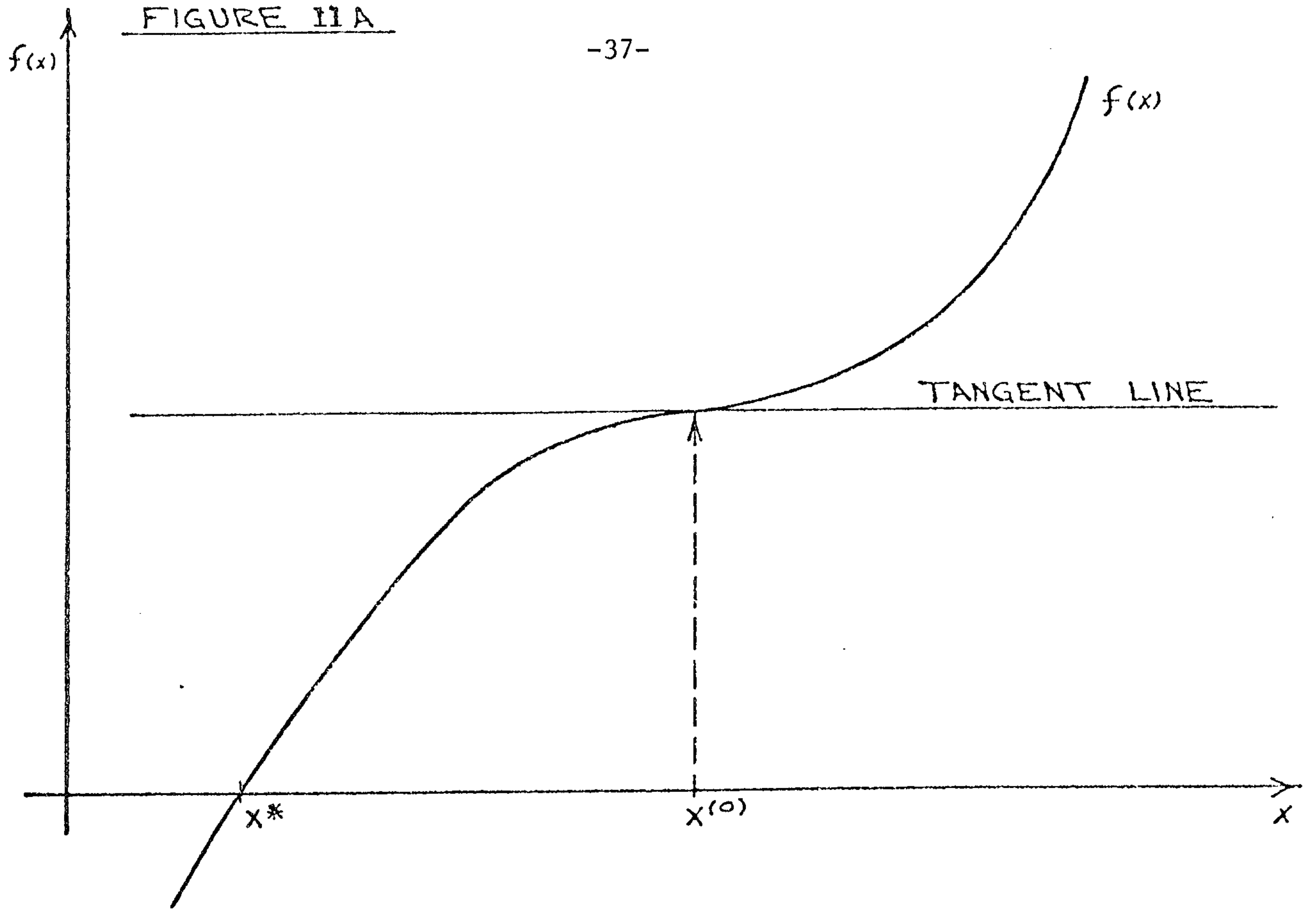
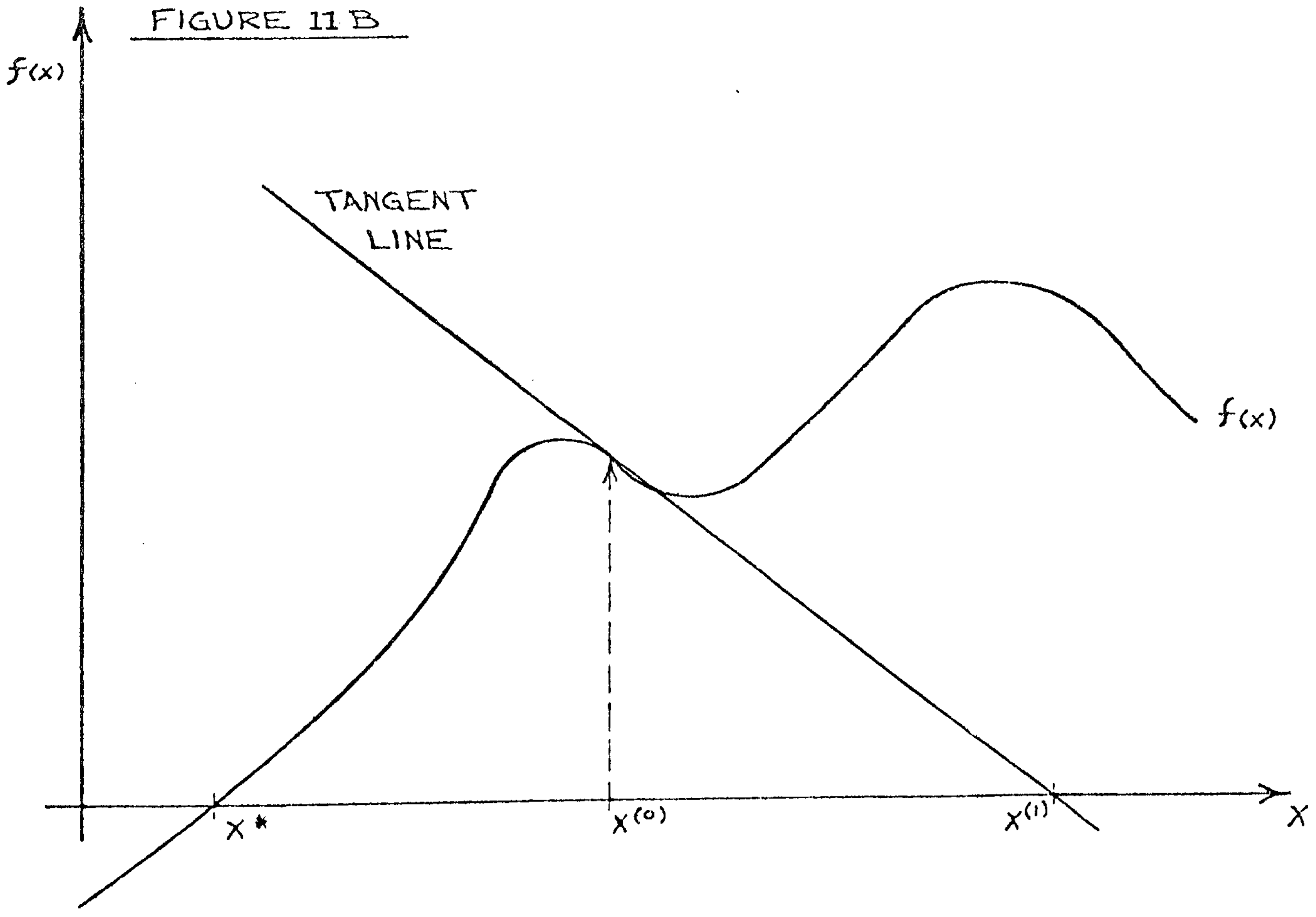


FIGURE 11B



divergence. Figure 11c shows a special case in which Newton's method neither converges nor diverges but rather cycles indefinitely between two points. In Figure 11d the sequence of iterates becomes negative before converging to the positive solution. Figure 11e illustrates the case when f(x) is not defined for all possible values of x. All of these cases have counterparts in higher dimensions. In higher dimensions there is not one function but n functions each with n first partial derivatives. Each function can be approximated at a given point by the hyperplane which is determined by its first partial derivatives at that point, or

$$\begin{aligned}
y_1 &= f_1(X^{(k)}) + (x_1 - x_1^{(k)}) \frac{\partial f_1}{\partial x_1}(X^{(k)}) + (x_2 - x_2^{(k)}) \frac{\partial f_1}{\partial x_2}(X^{(k)}) + \cdots + (x_n - x_n^{(k)}) \frac{\partial f_1}{\partial x_n}(X^{(k)}) \\
y_2 &= f_2(X^{(k)}) + (x_1 - x_1^{(k)}) \frac{\partial f_2}{\partial x_1}(X^{(k)}) + (x_2 - x_2^{(k)}) \frac{\partial f_2}{\partial x_2}(X^{(k)}) + \cdots + (x_n - x_n^{(k)}) \frac{\partial f_2}{\partial x_n}(X^{(k)}) \\
\vdots \quad & \quad \vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
y_n &= f_n(X^{(k)}) + (x_1 - x_1^{(k)}) \frac{\partial f_n}{\partial x_1}(X^{(k)}) + (x_2 - x_2^{(k)}) \frac{\partial f_n}{\partial x_2}(X^{(k)}) + \cdots + (x_n - x_n^{(k)}) \frac{\partial f_n}{\partial x_n}(X^{(k)}),
\end{aligned}$$

where $\frac{\partial f_i}{\partial x_j}(X^{(k)})$ is the j^{th} first partial derivative of the i^{th} equation of the system $F(X) = \underline{0}$ evaluated at the point $X^{(k)} = (x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$. These n tangent hyperplanes form an n x n system of linear equations that can be expressed in terms of the Jacobian, J, of $F(X) = \underline{0}$, or

$$Y = F(X^{(k)}) + J(X^{(k)})\,(X - X^{(k)}),$$

where

FIGURE 11C

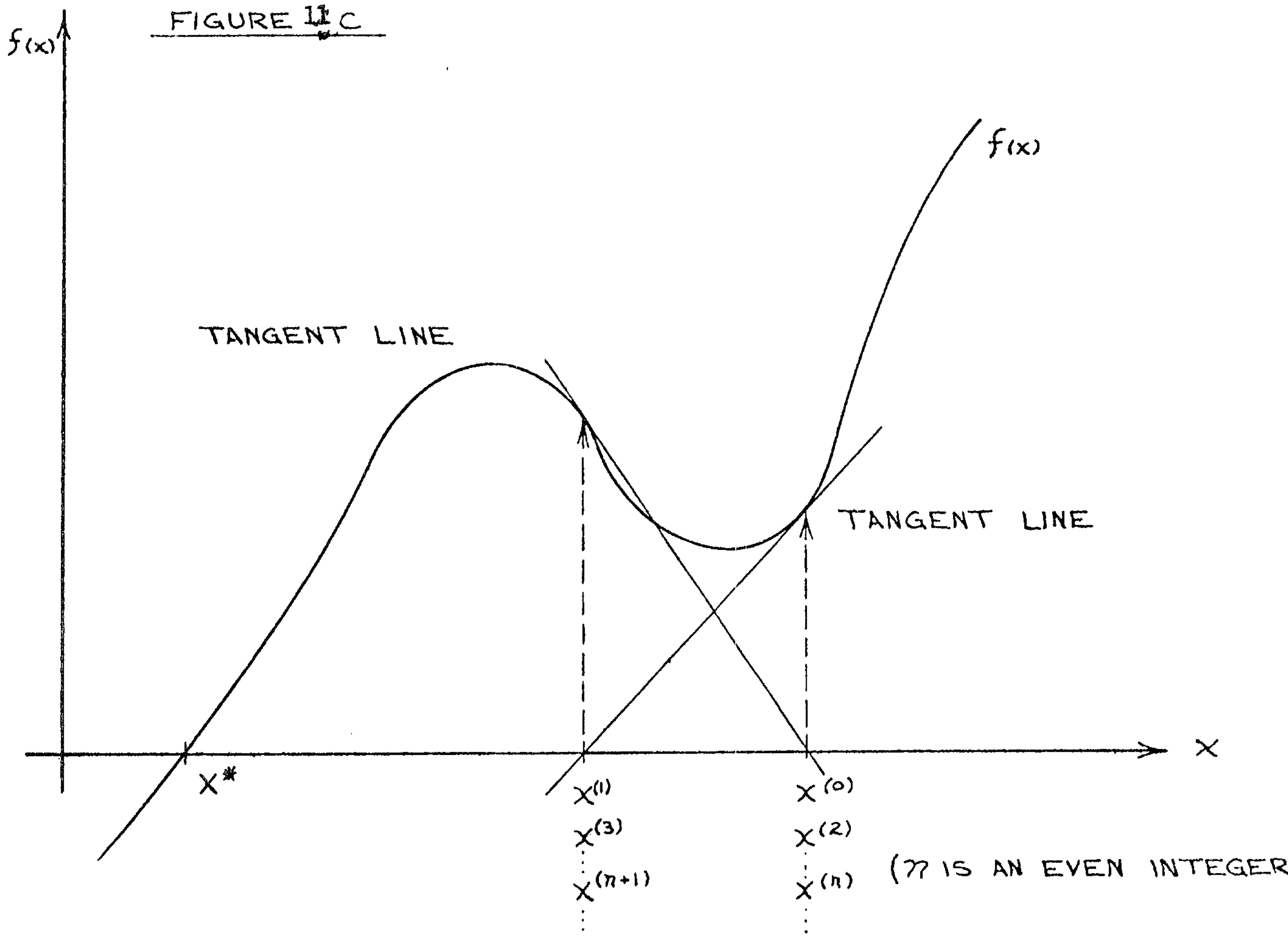
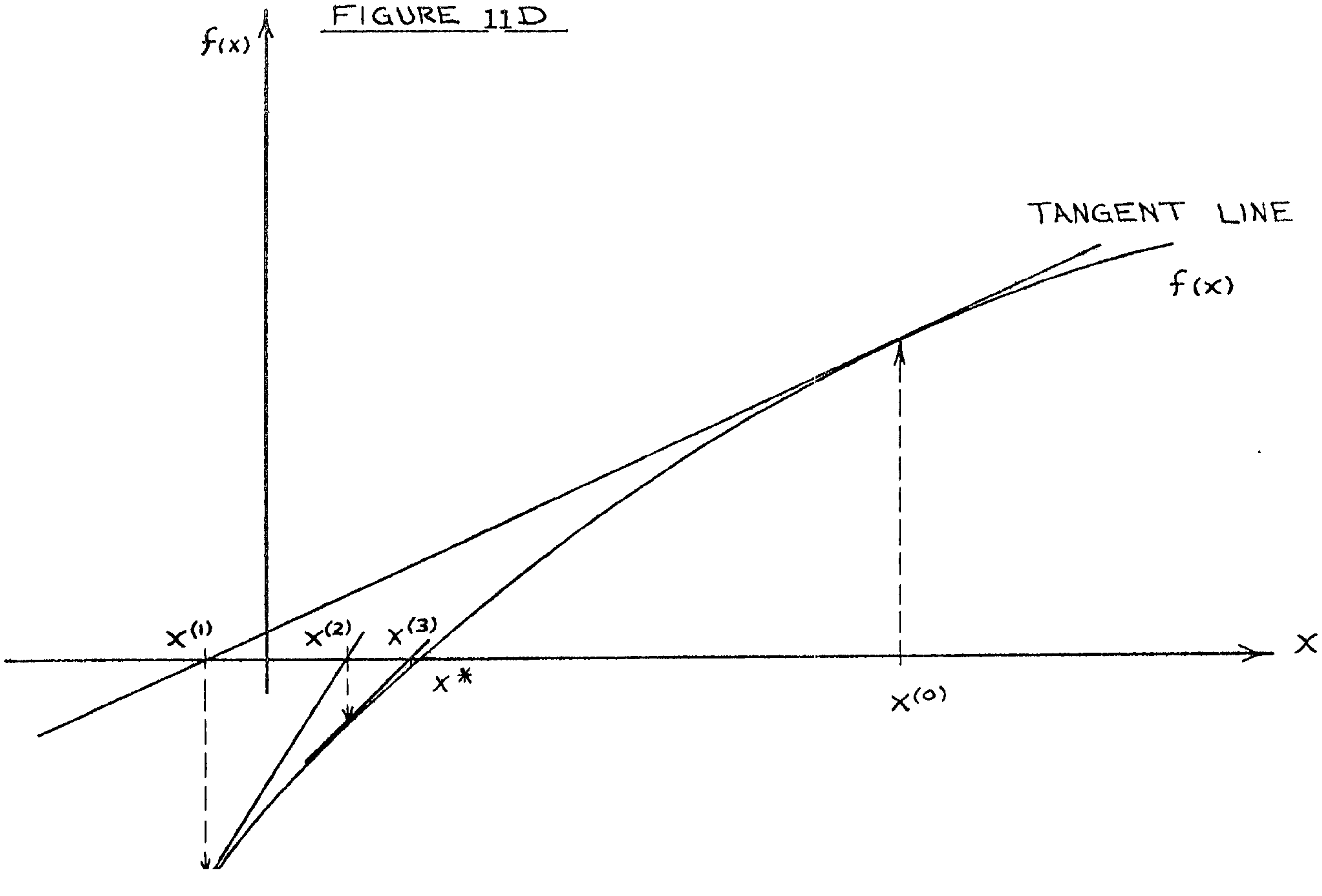


FIGURE 11D



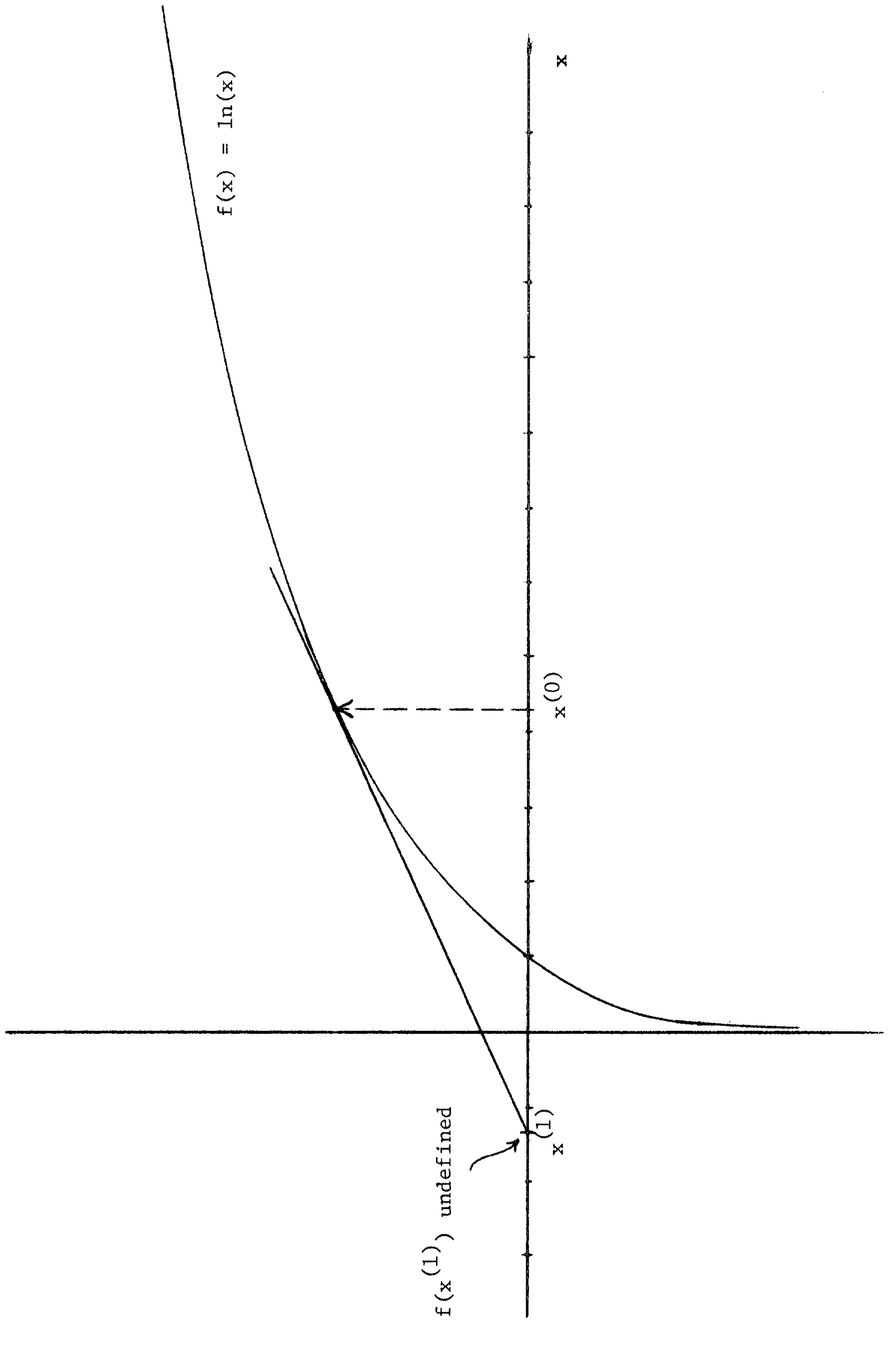


Figure 11E

$$J(X^{(k)}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(X^{(k)}) & \frac{\partial f_1}{\partial x_2}(X^{(k)}) & \cdots & \frac{\partial f_1}{\partial x_n}(X^{(k)}) \\ \frac{\partial f_2}{\partial x_1}(X^{(k)}) & \frac{\partial f_2}{\partial x_2}(X^{(k)}) & \cdots & \frac{\partial f_2}{\partial x_n}(X^{(k)}) \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(X^{(k)}) & \frac{\partial f_n}{\partial x_2}(X^{(k)}) & \cdots & \frac{\partial f_n}{\partial x_n}(X^{(k)}). \end{pmatrix}$$

Setting $Y = \underline{0}$ and solving for X to determine the new $X^{(k+1)}$ yields

$$J(X^{(k)})\,(X^{(k)}) - F(X^{(k)}) = J(X^{(k)})\,(X), \text{ or}$$

$$X^{(k+1)} = X^{(k)} - J^{-1}(X^{(k)})\,F(X^{(k)}).$$[17]

Thus $X^{(k+1)}$ is the intersection of the n tangent hyperplanes and the hyperplane $F(X) = \underline{0}$.

Each Newton iteration requires the calculation of the Jacobian of $F(X) = \underline{0}$ at the point $F(X^{(k)})$ and the solution of the resulting system of linear equations. Both of these processes are complicated when compared to a Jacobi or Gauss-Seidel iteration and require substantially more computer time and storage space per iteration. Frequently the derivatives are not explicitly evaluated, it being easier to approximate the first partial derivatives at $F(X^{(k)})$ by difference quotients like

$$\frac{\partial f_i}{\partial x_j}(X^{(k)}) = \left(\frac{1}{h}\right)\left[fi(x_1^{(k)}, \ldots, x_{i-1}^{(k)}, x_i^{(k)} + h, x_{i+1}^{(k)}, \ldots, x_n^{(k)}) - f_i(x_1^{(k)}, \ldots, x_n^{(k)})\right],$$

where i, j = 1, ..., n and h is a "small" positive number called a "discretization parameter." In this way the function is evaluated at n^2 points in the neighborhood of $X^{(k)}$ and the average rate of change between $F(X^{(k)})$ and these n^2 other points on $F(X)$ determines the n^2 elements of the approximately calculated Jacobian, $J(X^{(k)})$. There are variants to this technique, but the idea is always to avoid explicit calculation of the first partial derivatives. Because these methods can be included in a computer program for implementing a Newton-based method, it is not necessary for the user of the program to supply algebraic expressions for the partial derivatives of the system $F(X) = \underline{0}$. Newton methods employing these approximate derivatives are referred to as "discretized" Newton methods, but the distinction between exact and discretized Newton methods will not be maintained hereafter in this paper.

Once the Jacobian has been calculated there remains the problem of solving the resulting system of linear equations. A very serious difficulty arises when the Jacobian is singular. As in the one-dimensional case where $f'(x^{(k)}) = 0$, singularity of the Jacobian implies that no solution can be found to the system of linear equations,

$$\underline{0} = F(X^{(k)}) + J(X^{(k)})\ (X-X^{(k)}).$$

Newton's method is then undefined. This problem is discussed in more detail in the next section.

When the Jacobian is nonsingular a secondary algorithm must be supplied to solve the system of linear equations. Matrix inversion techniques could be used but commonly a secondary iterative method, such as the Gauss-Seidel or Jacobi, is used. This further increases the number of calculations in each Newton iteration. Offsetting the complexity of each Newton iteration is the fact that Newton-based methods typically require fewer iterations to achieve convergence than either the Jacobi or Gauss-Seidel methods. In fact, in the neighborhood of a solution X^*, Newton-based methods typically add two significant digits to the iterate in each iteration (this is referred to as "quadratic convergence"), while the Jacobi or Gauss-Seidel add no more than one significant digit per iteration (linear convergence) when $X^{(k)}$ is near X^*.

Brown's method is a Newton-based method that achieves some savings in computer storage space without sacrificing the rapid rate of convergence per iteration associated with the Newton method. The strategy of the method is again to approximate functions by tangent hyperplanes, but the resulting linear equations are treated sequentially rather than simultaneously. The method proceeds as follows. The first equation of the model, $f_1(X)$, is approximated by the tangent hyperplane[18] through the point $f_1(X^{(k)})$, that is, by

$$y_1 = f_1(X^{(k)}) + (x_1-x_1^{(k)})\, f_{11}(X^{(k)}) + (x_2-x_2^{(k)})\, f_{12}(X^{(k)}) + \cdots + (x_n-x_n^{(k)})\, f_{1n}(X^{(k)}),$$

where f_{ij} is an approximation of the j th first partial derivative of the i th equation. The intersection of this tangent plane with the hyperplane $F(X) = \underline{0}$ is expressed by setting $y_1 = 0$. Then, the variable x_i with the first partial derivative of largest magnitude ($f_{1i} > f_{1j}$, for all $j \neq i$) is expressed as a linear function of the other x_j's, $j \neq i$, or

$$x_i = L_i(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n).$$

Normalizing on the variable with first partial derivative of largest magnitude has two justifications. First, it prevents division by zero in most cases [2]. Secondly, it reduces the rounding errors introduced in computer calculations when division by a small number occurs [2].

Next, the second equation of the original system $F(X) = \underline{0}$, $f_2(x_1,\ldots,x_n)$, is rewritten with the function L_i substituting for x_i. It thus becomes a function of n-1 variables.

$$f_2(x_1, \ldots, x_n) = h_2(x_1, x_2, \ldots, x_{i-1}, L_i, x_{i+1}, \ldots, x_n).$$

As before, this nonlinear function h_2 is replaced by a linear approximation, which in this case is the n-1 dimensional hyperplane formed by the n-1 first partial derivatives of h_2. In order to derive another linear expression for some x_j, $j \neq i$, this linear expression is again set to zero and solved for the x_j with first partial derivative of largest magnitude, so that

$$x_j,\ j \neq i = L_j(x_1, \ldots, x_{i-1}, L_i, x_{i+1}, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n).$$

The expression L_j is then substituted back into L_i, reducing both L_i and L_j to linear functions of n-2 variables. Finally, the new L_i and L_j are then substituted into $f_3(x_1, x_2, \ldots x_n)$, and the process continues. Note that at each stage the new expressions L_k, etc., are substituted into the previous linearization, L_i, L_j, etc. Thus both the linear approximating functions and the remaining equations of the system are reduced in dimension at each step. This is what allows for the saving of computer storage space.

The method proceeds through each equation one by one until the last equation is expressed as a function of one variable, x_ℓ, and n-1 linearizations, $L_1, L_2, \ldots L_{\ell-1}, L_{\ell+2}, \ldots, L_n$. This function is linearized, set equal to zero, and an exact numerical value for $x_\ell^{(k+1)}$ determined. This value is substituted back into the n-1 one-dimensional linear expressions, L_i, $i \neq \ell$, and new values are calculated for

$$x_1^{(k+1)}, \ldots, x_{\ell-1}^{(k+1)}, x_{\ell+1}^{(k+1)}, \ldots, x_n^{(k+1)}.$$

The following example explicitly illustrates these procedures:

Consider $F(X) = \underline{0}$, where F is the 3 x 3 system of simultaneous equations.

$$f_1(X) = f_1(x_1, x_2, x_3) = 0$$

$$f_2(X) = f_2(x_1, x_2, x_3) = 0$$

$$f_3(X) = f_3(x_1, x_2, x_3) = 0$$

Given an iterate $X^{(k)} = (x_1^{(k)}, x_2^{(k)}, x_3^{(k)})$, solution by Brown's method proceeds as follows.

(1) Approximate $f_1(X)$ by a tangent plane through $f_1(X^{(k)})$,

$$0 = f_1(X^{(k)}) + (x_1 - x_1^{(k)}) \frac{\partial f_1}{\partial x_1}(X^{(k)}) + (x_2 - x_2^{(k)}) \frac{\partial f_1}{\partial x_2}(X^{(k)}) + (x_3 - x_3^{(k)}) \frac{\partial f_1}{\partial x_3}(X^{(k)})$$

(2) **Assuming, for example, that**

$$\frac{\partial f_1}{\partial x_2}(X^{(k)}) > \frac{\partial f_1}{\partial x_1}(X^{(k)}) \text{ and } \frac{\partial f_1}{\partial x_2}(X^{(k)}) > \frac{\partial f_1}{\partial x_3}(X^{(k)}),$$

solve for x_2 in terms of x_1 and x_3.

$$x_2 = \frac{\left[-f_1(X^{(k)}) + x_2^{(k)} \frac{\partial f_1}{\partial x_2}(x_1^{(k)}) + x_1^{(k)} \frac{\partial f_1}{\partial x_1}(X^{(k)}) + x_3^{(k)} \frac{\partial f_1}{\partial x_3}(X^{(k)})\right].}{\frac{\partial f_1}{\partial x_2}(X^{(k)})}$$

$$-(x_3)\,\frac{\frac{\partial f_1}{\partial x_3}\,(X^{(k)})}{\frac{\partial f_1}{\partial x_2}\,(X^{(k)})} - (x_1)\,\frac{\frac{\partial f_1}{\partial x_1}\,(X^{(k)})}{\frac{\partial f_1}{\partial x_2}\,(X^{(k)})}$$

Setting $L_2 = x_2$ and replacing constant terms by letters from the beginning of the alphabet yields

$$L_2 = \frac{c}{d} - \frac{a}{d}\,x_3 - \frac{b}{d}\,x_1$$

(3) Substitute L_2 for x_2 in $f_2(X^{(k)})$ and repeat the steps above.

(a) substitution

$$f_2(x_1^{(k)},\, x_2^{(k)},\, x_3^{(k)}) = h_2(x_1^{(k)},\, L_2\;,\, x_3^{(k)})$$

(b) approximation--let ( ) denote $(x_1^{(k)},\, L_2\;,\, x_3^{(k)})$.

$$0 = h_2\,(\;) + (x_1 - x_1^{(k)})\,\frac{\partial h_2}{\partial x_1}\,(\;) + (x_3 - x_3^{(k)})\,\frac{\partial h_2}{\partial x_3}\,(\;)$$

$$+ (x_1 - x_1^{(k)})\,\frac{\partial h_2}{\partial L_2}\,(\;)\,\frac{\partial L_2}{\partial x_1}\,(\;) + (x_3 - x_3^{(k)})\,\frac{\partial h_2}{\partial L_2}\,(\;)\,\frac{\partial L_2}{\partial x_3}\,(\;)$$

$$= h_2\,(\;) + (x_1 - x_1^{(k)})\,\Big[\frac{\partial h_2}{\partial x_1}\,(\;) - \frac{b}{d}\,\frac{\partial h_2}{\partial L_2}\,(\;)\Big]$$

$$+ (x_3 - x_3^{(k)}) - \Big[\frac{\partial h_2}{\partial x_3}\,(\;) - \frac{a}{d}\,\frac{\partial h_2}{\partial L_2}\,(\;)\Big]$$

Replacing constant terms by letters from the middle of the alphabet yields

$$0 = \ell + (x_1 - x_1^{(k)})\, m + (x_3 - x_3^{(k)})\, p$$

(c) Assume, for example, $m > p$ and solve for x_1 in terms of x_3; call it L_1.

$$L_1 = \frac{-\ell}{m} + x_1^{(k)} - \frac{p}{m} x_3 + \frac{p}{m} x_3^{(k)} = r - \frac{p}{m} x_3,$$

where $r = \frac{-\ell}{m} + \frac{p}{m} x_3^{(k)}$.

(4) Substitute L_1 for x_1 in L_2

$$L_2 = \frac{c}{d} - \frac{a}{d} x_3 - \frac{b}{d} L_1$$

(5) Substitute L_1 and L_2 in $f_3(X^{(k)})$ and solve for x_3 by the same procedure as above.

(a) substitution

$$f_3(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}) = h_3(L_1, \quad L_2, \quad x_3^{(k)})$$

(b) approximation--let ( ) denote $(L_1, \quad L_2, \quad x_3^{(k)})$.

$$0 = h_3\,(\;) + (x_3 - x_3^{(k)}) \frac{\partial h_3}{\partial x_3}(\;) + (x_3 - x_3^{(k)}) \frac{\partial h_3}{\partial L_1}(\;) \frac{\partial L_1}{\partial x_3}(\;)$$

$$+ (x_3 - x_3^{(k)}) \frac{\partial h_3}{\partial L_2}(\;) \frac{\partial L_2}{\partial x_3}(\;) = h_3\,(\;) + (x_3 - x_3^{(k)})$$

$$+ \left[\frac{\partial h_3}{\partial x_3}(\;) - \frac{p}{m} \frac{\partial h_3}{\partial L_1}(\;) - \frac{a}{d} \frac{\partial h_3}{\partial L_2}(\;) \right]$$

Replacing constant terms by letters of the alphabet yields

$$0 = s + (x_3 - x_3^{(k)})\, t$$

(c) Solve for x_3, call it $x_3^{(k+1)}$

$$x_3 = \frac{-s}{t} + x_3^{(k)} = x_3^{(k+1)}$$

(6) Substitute $x_3^{(k+1)}$ for x_3 in L_1 and L_2 to determine $x_1^{(k+1)}$ and $x_2^{(k+1)}$

$$L_1 = r - (\frac{P}{m})\, x_3,\ 50\ x_1^{(k+1)} = r - (\frac{P}{m})\, x_3^{(k+1)}$$

$$L_2 = \frac{c}{d} - \frac{b}{d} L_1 - \frac{a}{d} x_3,\ 50\ x_2^{(k+1)} = \frac{c}{d} - \frac{b}{d}\,(r - (\frac{P}{m})\, x_3^{(k+1)}) - (\frac{a}{d})\, x_3^{(k+1)}$$

$$= (\frac{c}{d} - \frac{br}{d}) + (\frac{bp}{dm} - \frac{a}{d})\, x_3^{(k+1)} .$$

This completes an iteration of Brown's method from $X^{(k)}$ to $X^{(k+1)}$ on the 3 x 3 simultaneous system $F(X) = \underline{0}$.

Brown's method reduces the amount of storage space and number of calculations per iteration by a factor of $(n^2+3n/2n^2+2n)$ [2] compared to Newton's method. This is a significant savings in large systems but still far exceeds the complexity per iteration typical of Jacobi or Gauss-Seidel methods.

F. Implementing the Newton and Brown Methods

Objections frequently expressed against Newton-based methods are that they require large amounts of computer storage and become undefined when the Jacobian is singular at some iterate [12]. New variants, such as Brown's method, achieve some savings in storage space, but all methods involving linearization by first partial derivatives will require much more storage space than Jacobi or Gauss-Seidel methods. However, given the capacity of modern computers, the storage requirements of Newton-based methods are not a serious obstacle to their use on most economic models. The problem of sigularity is also rarely of practical concern. The following suggestions should help achieve convergence with the Newton or Brown method.

1. Picking an initial guess. The recommendations are identical to those for the Gauss-Seidel and Jacobi methods (see p. 24).

2. Ordering and normalization of equations. In Newton's method the linear approximating functions are generated and solved simultaneously. Barring divergence of the secondary algorithm used to solve the approximating linear system, ordering and normalization of equations is not crucial to the convergence of Newton's method.

In Brown's method, however, the equations are linearized one by one, as explained above. Because of the sequential use of the equations it is desirable to list the "most linear" equations first. This insures that the initial linear approximations, which will be passed on by substitution to all the remaining equations, will be reasonably accurate. Relative linearity is determined by the degree of the highest polynomial term in the equation.19/ For example, the equation $x_1^2 + 3x_2^3+x_5 = 0$ is more linear than and would be listed before the equation $x_3^4 = 0$.

Though a good rule of thumb, ordering the equations by decreasing linearity does not guarantee an optimal rate of convergence for Brown's method in all cases. A superior ordering, when circumstances permit, is to arrange the equations so that the Jacobian matrix is dominant diagonal (i.e., so that $J_{ii} > \sum_{j\neq i} J_{ij}$, i=1, ..., n).

3. Singularly of the Jacobian.--Often considered a major objection to a Newton-based method,20/ singularity is neither a common nor an

insurmountable difficulty in practice.[21/] The authors have never encountered it while using Brown's method to solve a simultaneous equation economic model, and other users[22/] of Brown's and Newton's methods have privately communicated their similar experiences. Singularity may be more troublesome in other applications[23/], however. When singularity is encountered, procedures are available for selecting an alternative iterate. The variant of Brown's method in the IMSL library contains one procedure. Another has been developed by David M. Gay [9].

4. Dampening. Dampening in Newton's or Brown's method is similar to dampening in Jacobi or Gauss-Seidel methods. The new iterate, $x^{(k+1)}$, is a weighted average of the old iterate, x(k), and the iterate that would have been chosen by the original, undampened method. In the dampened Newton's method, for example, $X_{DN}^{(k+1)} = \delta X^{(k)} + (1-\delta)X_N^{(k+1)}$, where subscripts DN and N indicate the iterates chosen by the dampened Newton's method and ordinary Newton's method, respectively. Since $X_N^{(k+1)} = X^{(k)} - J^{-1}(X^{(k)})F(X^{(k)})$, the dampened Newton's method is often rewritten as $X_{DN}^{(k+1)} = \delta X^{(k)} + (1-\delta)[X^{(k)} - J^{-1}(X^{(k)})F(X^{(k)})] = X^{(k)} + \delta J^{-1}(X^{(k)})F(X^{(k)})$. Variable dampening factors, where δ is a function of $F(X^{(k)})$, have also been used [10, pp. 46-47].

The effect of dampening is to choose a new iterate in the direction picked by the undampened method but at a distance controlled by the dampening factor. Usually $0 < \delta < 1$ and the new iterate is closer to $X^{(k)}$ than the new iterate that would be chosen by the undampened method. This "conservative" approach can be very useful for solving systems of equations which are "well-behaved" on one side of a root but not on the other. Figure 12a shows a function which is positively sloped for all X greater than the root X* but first positively and then negatively sloped for X < X*. Beginning at $X^{(1)}$, the undampened Newton's method chooses $X^{(2)}$

Figure 12a Without Dampening

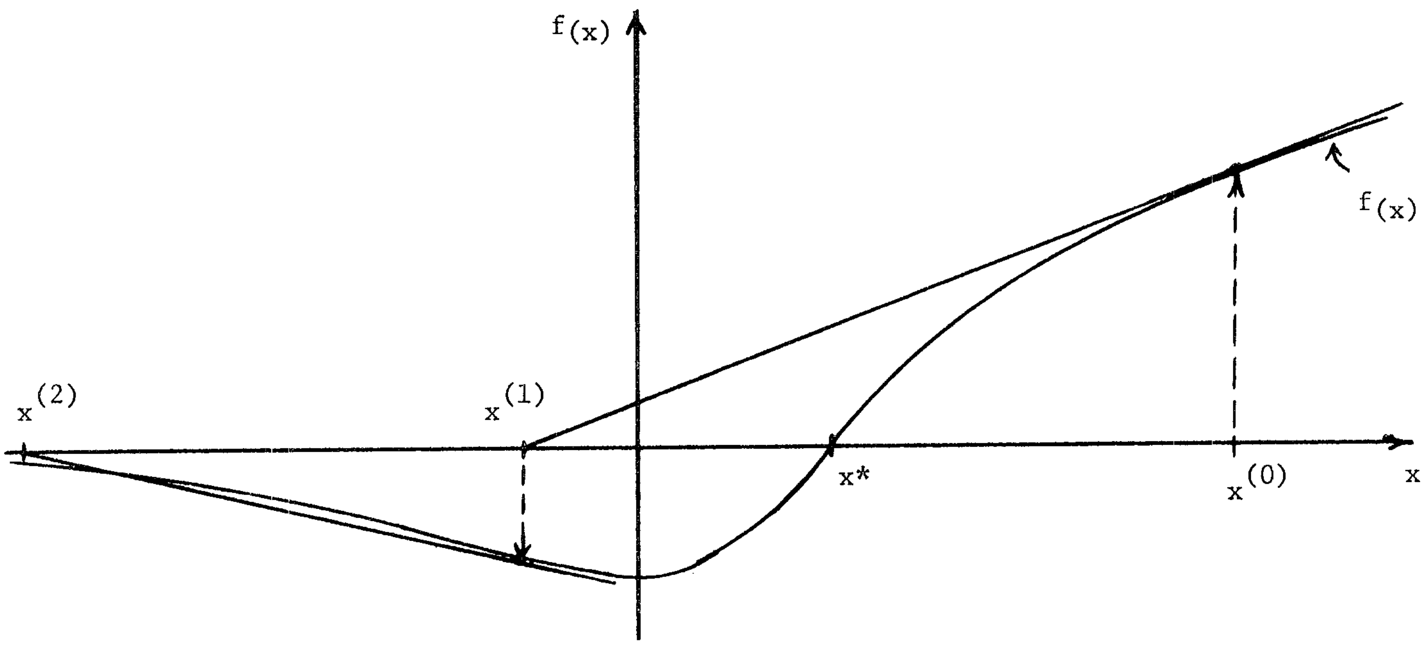
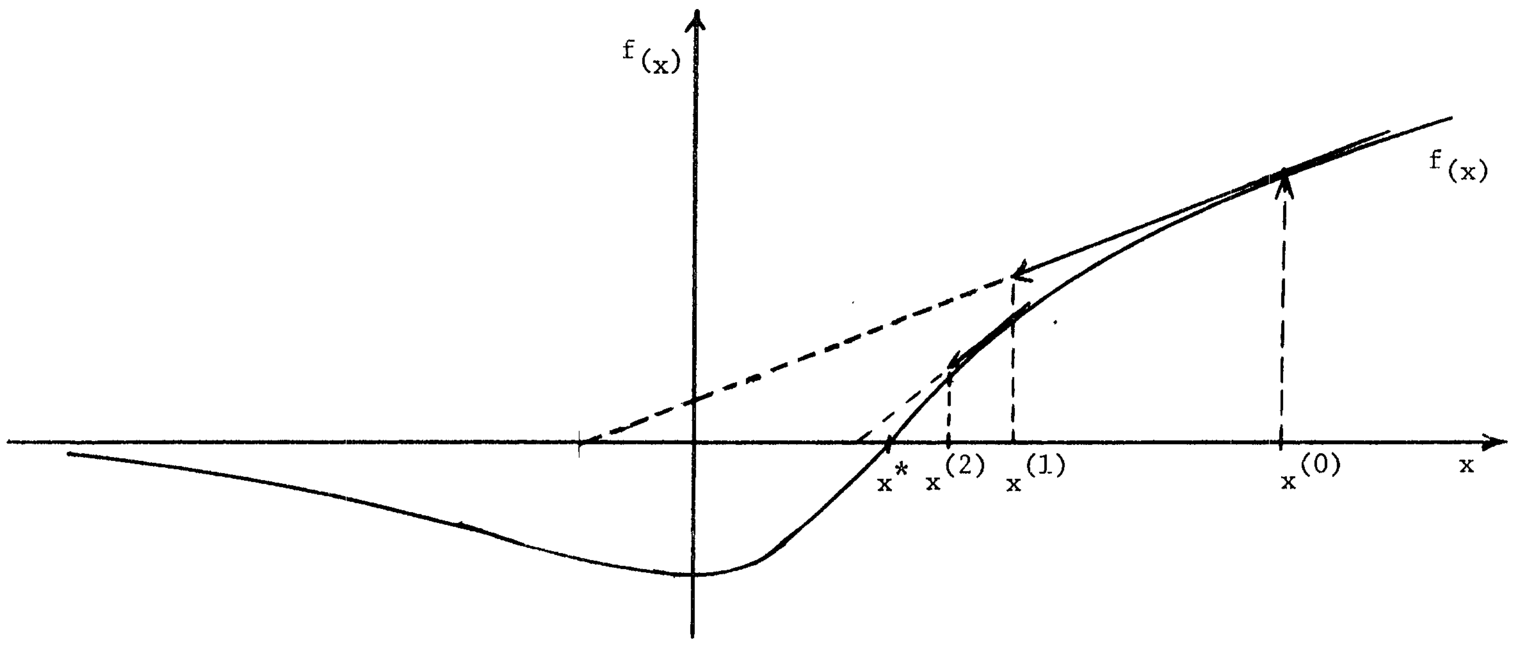


Figure 12b With Dampening



in the correct direction (leftward). $X^{(2)}$, however, lies in the portion of X-axis where F(X) is negatively sloped. New iterates will be chosen further and further to the left of X*. The problem in this case seems to arise when the choice of $X^{(2)}$ overshoots the portion of the x-axis where the undampened Newton's method chooses the correct direction. The same example is repeated in Figure 12b except that a dampening factor is used to limit the changes in $X^{(k)}$. Convergence is achieved because all the successive iterates lie within the region about X* where F(X) is smooth and positively sloped. Dampening factors can also be useful when F(X) is only defined to one side of X* or in some region around X*. The logarithmic function shown in Figure 11e is a good example.

Part IV. Concluding Remarks

A. General Considerations

Newton's method, Brown's method, and the Gauss-Seidel and Jacobi methods can be adapted to provide solutions to most economic models. This may be more difficult for larger models, as the number of ways to order or normalize the equations increases rapidly. For this reason it is sometimes necessary to initially solve the model in separate blocks before applying the algorithm to the entire model. It is advisable to begin using iteration in the early stages of the model building process. Experience gained at the outset, when models are typically smaller, simpler, and easier to solve, may help considerably in preparing the final model for iterative solution.

B. Selecting an Iterative Method

The four iterative methods described in Parts II and III are all suitable for solving economic models composed of n equations in n unknowns. Many other iterative methods[25/] could also be used. There is no single best method to recommend, and much practical information about current iterative methods is only available by word of mouth. Time can be saved by asking for help before writing and running a program. Advice should be sought from a computer scientist or numerical analyst if one or several models must be solved repeatedly, and especially if the models

In many cases one of the four methods described above will be adequate to solve the economic model. Experience with nonlinear simultaneous-equation economic models at the Department of Agricultural and Applied Economics of the University of Minnesota supports the following conclusions:

1. The dampened Jacobi and dampened Gauss-Seidel methods are more likely to converge than the corresponding undampened methods.
2. The dampened Jacobi method solves economic models more regularly than the dampened Gauss-Seidel method.
3. The dampened Jacobi method is especially suited to solving simultaneous-equation econometric models.
4. The dampened Jacobi method is very competitive with Brown's or Newton's method. In a series of tests on three simultaneous equation economic models, the dampened Jacobi method solved the models at least as frequently as Brown's method and required from slightly more to significantly fewer seconds of computer time. However, in one case the Jacobi method failed to solve the model as initially normalized and had to be renormalized [21, pp. 52-59].

The dampened Jacobi method, Brown's method, and Newton's method are all currently being used to solve simultaneous-equation economic models and can be recommended to other researchers for this purpose. Use of the methods for other purposes (e.g., solving first-order conditions of maximization of likelihood functions) is not ruled out, but for many other problems there exist distinct types of competing solution procedures (e.g., algorithms for constrained optimization).

APPENDIX A: Computer Programs for Implementing Brown's Method and the Dampened Jacobi method.

Fortran programs used to implement Brown's method and the dampened Jacobi method on a specific model are presented here to clarify some of the practical aspects of using these techniques. The model used is a modified version of the 14-equation dualistic growth model developed by Kelly, Williamson, and Cheetam [15; pp. 22-57]. The model consists of the following set of equations:

Production Relationships

1) $Q_1 = A_1[K_1^{((\sigma_1 - 1)/\sigma_1)} + L_1^{((\sigma_1 - 1)/\sigma_1)}]^{(V_1\sigma_1/(\sigma_1 - 1))}$

2) $Q_2 = A_2[K_2^{((\sigma_2 - 1)/\sigma_1)} + L_2^{((\sigma_2 - 1)/\sigma_2)}]^{(V_2\sigma_2/(\sigma_2 - 1))}$

Factor Demands

3) $w = P \cdot V_1 \cdot A_1^{((\sigma_1 - 1)/V_1\sigma_1)} \cdot L_1^{(-1/\sigma_1)} \cdot Q_1^{[(\sigma_1(V_1-1)+1)/V_1\sigma_1]}$

4) $w = V_2 \cdot A_2^{((\sigma_2 - 1)/V_2\sigma_2)} \cdot L_2^{(-1/\sigma_2)} \cdot Q_2^{[(\sigma_2(V_2-1)+1)/V_2\sigma_2]}$

5) $r = P \cdot V_1 \cdot A_1^{((\sigma_1 - 1)/V_1\sigma_1)} \cdot K_1^{(-1/\sigma_1)} \cdot Q_1^{[(\sigma_1(V_1-1)+1)/V_1\sigma_1]}$

6) $r = V_2 \cdot A_2^{((\sigma_2-1)/V_2\sigma_2)} \cdot K_2^{(-1/\sigma_2)} \cdot Q_2^{[(\sigma_2(V_2-1)+1)/V_2\sigma_2]}$

Commodity Demands

7) $D_{11} = (L_1/P) \cdot B_{11} \cdot (w - G)$

8) $D_{12} = (L_2/P) \cdot B_{12} \quad (w - G)$

9) $D_{21} = L_1(G + B_{21}(w - G))$

10) $D_{22} = L_2(G + B_{22}(w - G))$

Investment

11) $I = (r \cdot C - XM)/P$

Full Employment

12) $C = K_1 + K_2$

13) $L = L_1 + L_2$

Market Balance

14) $Q_1 = D_{11} + D_{12} + I$

15) $Q_2 = D_{21} + D_{22}$.

The variables are defined as follows:

Endogenous

6) Q_1 = output in the first (industrial) sector

7) Q_2 = output in the second (agricultural) sector

3) L_1 = laborers employed in the first sector

4) L_2 = laborers employed in the second sector

1) K_1 = capital employed in the first sector

2) K_2 = capital employed in the second sector

5) P = price of industrial goods in terms of agricultural goods

8) w = returns (wages) to a unit of labor

9) r = returns (rent) to a unit of capital

10) I = investment, the amount of Q_1 bought for other than consumption purposes

11) D_{11} = the amount of Q_1 consumed by laborers in the industrial sector

12) D_{12} = the amount of Q_1 consumed by laborers in the agricultural sector

13) D_{21} = the amount of Q_2 consumed by laborers in the industrial sector

14) D_{22} = the amount of Q_2 consumed by labors in the agricultural sector.

Exogenous, or Predetermined

C	= total amount of capital available
L	= total amount of labor available
V_1, V_2	= the degree of homogeneity in the constant elasticity of substitution (CES) production functions for Q_1 and Q_2
σ_1, σ_2	= the elasticity of substitution of labor for capital in the CES production functions for Q_1 and Q_2.
G	= the minimum or subsistence amount of food which each laborer must consume to sustain life
XM	= the aggregate amount of food necessary to sustain the owners of capital
B_{11}	= the proportion of discretionary income (income remaining after satisfaction of the subsistence food consumption requirement) spent on industrial goods by industrial laborers.
B_{12}	= the proportion of discretionary income spent on industrial goods by agricultural laborers.
B_{21}	= the proportion of discretionary income spent on agricultural goods by industrial laborers
B_{22}	= the proportion of discretionary income spent on agricultural goods by agricultural laborers.

Finally the values assigned to the exogenous variables were the following:

$C = 30.0$

$L = 100.0$

$V_1 = V_2 = 1.0$

$\sigma_1 = 0.5$

$\sigma_2 = 1.5$

$G = 0.648$

$XM = 0.0$

$B_{11} = 0.8$

$B_{12} = 0.5$

$B_{21} = 0.2$

$B_{22} = 0.5$

The reader should note that the model as just stated is not in a form compatible with any of the iterative methods discussed in this paper. The model is not normalized in the form suitable for Jacobi or Gauss-Seidel iteration; neither is it written in the implicit function form required for Brown's or Newton's method. More fundamentally, the model has 15 equations but only 14 endogenous variables. If all 15 equations were independent the model could be inconsistent and have no solution. Economic theory must be used to discover a "square" model (one with an equal number of equations and endogenous variables, an n x n system) having the same solution as the original model. Otherwise, iterative techniques cannot be applied. In the Kelley-Williamson-Cheetam model, "if we assume that there is at least one positive value for the terms of trade [P] that will satisfy the equilibrium conditions, it is a simple matter to show that equilibrium in the two factor markets and in either commodity market necessarily implies equilibrium in the remaining commodity market. Thus, one of the two commodity market equations can be ignored; the model then becomes a system of fourteen equations and fourteen variables [15, p. 51]." Equation 15 will henceforth be omitted from the model without affecting the solution values.

1. Programming Brown's Method.--Brown's method is available as a library subroutine NONLIN on the University of Minnesota computer system. A similar version is also available at many locations in the International Mathematics and Statistics Library (IMSL), under the name ZSYSTM. To implement either of these algorithms the user must first rewrite the model in implicit function form. This can be accomplished by subtracting the right-hand side of each equation from the left-hand side. The next step is to set each implicit function equal to an arbitrary function name, "FF" for example, and to label the function with a statement number. The market-balance equations of the Kelley-Williamson-Cheetam model would thus be rewritten as follows:

12 $FF = C - K_1 - K_2$

13 $FF = L - L_1 - L_2$

14 $FF = Q_1 - D_{11} - D_{12} - I.$

However, the equations must also be written in matrix notation. Each endogenous variable must be designated as an element of an n-unit vector, which will be called X in this example. The elements of X will be as follows:

$X(1) = K_1$

$X(2) = K_2$

$X(3) = L_1$

$X(4) = L_2$

$X(5) = P$

$X(6) = Q_1$

$X(7) = Q_2$

$X(8) = w$

$X(9) = r$

$X(10) = I$

$X(11) = D_{11}$

$X(12) = D_{12}$

$X(13) = D_{21}$

$X(14) = D_{22}$.

The market balance equations, in final form, are written

```
12  FF = C - X(1) - X(2)
13  FF = L - X(3) - X(4)
14  FF = X(6) - X(11) - X(12) - X(10).
```

When all the functions have been rewritten in this form they are arranged as in the function FF which appears in the program reproduced in Figure 10. The remainder of the program follows directly from the instructions accompanying the NONLIN or ZSYSTM packages. In Figure 13 the required statements which are problem specific (i.e., must be changed for each different model but cannot be omitted) are underlined. Cards pertaining only to the

FIGURE 13 - page i

```
                                                                              Statement Numbers
      PROGRAM T (OUTPUT, TAPE6=OUTPUT)                                                1
      DIMENSION WORK(147), X(14)                                                      2
      COMMON A,C,XL,TT,B41,B12,B22,B32,B42,B13,B23,B33,B43,B14,B24,B34,B            *  3
     144,B11,B21,B31,AL3,AL4,Z31,Z32,Z33,Z41,Z42,Z43,G                                4
      COMMON R2,S2,XM,AL2,S1,V1,V2,AL1,R1,E1,E2,E3,E4,GT                            *  5
      EXTERNAL FF,PRINT                                                               6
      N=14                                                                          *  7
C     INITIAL VALUES                                                                  8
      X(1)=18.174                                                                   *  9
      X(2)=12.0                                                                     * 10
      X(3)=42.7                                                                     * 11
      X(4)=60.0                                                                     * 12
      X(5)=11.0                                                                     * 13
      X(6)=10.0                                                                     * 14
      X(7)=95.0                                                                     * 15
      X(8)=0.9                                                                      * 16
      X(9)=2.7                                                                      * 17
      X(10)=2.0                                                                     * 18
      X(11)=2.0                                                                     * 19
      X(12)=2.0                                                                     * 20
      X(13)=21.0                                                                    * 21
      X(14)=45.0                                                                    * 22
C     PARAMETER VALUES                                                               23
      G=0.648                                                                       * 24
      XM=0.0                                                                        * 25
C     XL EQUIVALENT TO L, TOTAL LABOR                                               * 26
      XL=100.0                                                                      * 27
      C=30.0                                                                        * 28
      B12=0.5                                                                       * 29
      B11=0.8                                                                       * 30
      B21=0.2                                                                       * 31
      B22=0.5                                                                       * 32
C     AL1 EQUIVALENT TO A1                                                          * 33
      AL1=0.64                                                                      * 34
C     AL2 EQUIVALENT TO A2                                                          * 35
      AL2=0.35                                                                      * 36
C     S1 EQUIVALENT TO SIGMA 1                                                      * 37
      S1=0.5                                                                        * 38
      R1=(1.0-S1)/S1                                                                * 39
```

FIGURE 13 - page ii

```
                                                                    Statement Numbers
     S2 EQUIVALENT TO SIGMA 2                                             * 40
     S2=1.5                                                               * 41
     R2=(1.0-S2)/S2                                                       * 42
     V1=1.0                                                               * 43
     V2=1.0                                                               * 44
     GT=0.0                                                               * 45
     T1=CPTIME(1.0)                                                       * 46
     PRINT 10, T1                                                           47
     CALL NONLIN (FF,X,14,25,14,1.0E-8,WORK,IERR,0,PRINT)                   48
     T2=CPTIME(1.0)                                                         49
     PRINT 15, T2                                                           50
     SS=0.0                                                                 51
     DO 5 J=1,14                                                            52
       FT=FF(X,J)                                                           53
       SS=SS+FT*FT                                                          54
5      WRITE (6,20) J,X(J),FT                                               55
     WRITE (6,25) IERR,SS                                                   56
     STOP                                                                   57
C                                                                           58
10   FORMAT (30X,*T1=*,3X,F10.5)                                            59
15   FORMAT (30X,*T2=*,3X,F10.5)                                            60
20   FORMAT (5H I = ,I1,8H X(I) = ,E22.14,8H F(I) = ,E22.14)                61
25   FORMAT (//8H IERR = ,I2,13H SUMSQUARE = ,E22.14)                       62
     END                                                                    63
```

FIGURE 13- page iii

```
                                                                                Statement Numbers
      FUNCTION FF(X,J)                                                                  1
      DIMENSION X(14)                                                                   2
      COMMON A,C,XL,TT,B41,B12,B22,B32,B42,B13,B23,B33,B43,B14,B24,B34,B              * 3
     144,B11,B21,B31,AL3,AL4,Z31,Z32,Z33,Z41,Z42,Z43,G                                  4
      COMMON R2,S2,XM,AL2,S1,V1,V2,AL1,R1,E1,E2,E3,E4,GT                              * 5
C     CHOOSE THE ORDER OF EQUATIONS                                                     6
      GO TO (60,65,70,25,30,15,20,5,10,35,40,45,50,55), J                               7
C     STATEMENT OF THE MODEL IN IMPLICIT FUNCTION FORM                                  8
C     KELLEY-WILLIAMSON MODEL.                                                        * 9
C     1 INDICATES INDUSTRIAL, 2 INDICATES AGRICULTURAL SECTOR                         * 10
C     VARIABLE NAMES ARE X(1)=K1,X(2)=K2,X(3)=L1,X(4)=L2,X(5)=P,X(6)=Q1,              * 11
C  X(7)=Q2,X(8)=W,X(9)=R,X(10)=I,X(11)=D11,X(12)=D12,X(13 =D21,X(14)=D21,               12
C     EXOGENOUS VARIABLE NAMES C=K,XL=L,A=T                                           * 13
C     PRODUCTION RELATIONSHIPS                                                        * 14
5     FF=X(6)-AL1*(((X(1)**(-R1))+(X(3)**(-R1)))**(V1*S1/(S1-1.0)))                   * 15
      RETURN                                                                            16
10    FF=X(7)-AL2*((((X(2))**(-R2))+((X(4))**(-R2)))**(-1.0/R2))                      * 17
      RETURN                                                                            18
C     COMMODITY DEMAND                                                                  19
15    FF=X(11)-(X(3)/X(5))*B11*(X(8)-G)                                               * 20
      RETURN                                                                            21
20    FF=X(12)-(X(4)/X(5))*B12*(X(8)-G)                                               * 22
      RETURN                                                                            23
25    FF=X(13)-(X(3))*(G+B21*(X(8)-G))                                                * 24
      RETURN                                                                            25
30    FF=X(14)-X(4)*(G+B22*(X(8)-G))                                                  * 26
      RETURN                                                                            27
C     FACTOR DEMAND                                                                   * 28
35    FF=X(8)-V2*(AL2**((S2-1.0)/(V2*S2)))*(X(4)**(-1.0/S2))*(X(7)**((S2              * 29
     1*(V2-1.0)+1.0)/(V2*S2)))                                                          30
      RETURN                                                                            31
40    FF=X(9)-V1*(AL1**((S1-1.0)/(V1*S1)))*(X(1)**(-1.0/S1))*(X(6)**((S1              * 32
     1*(V1-1.0)+1.0)/(V1*S1)))*X(5)                                                     33
      RETURN                                                                            34
45    FF=X(9)-V2*(AL2**((S2-1.0)/(V2*S2)))*(X(2)**(-1.0/S2))*(X(7)**((S2              * 35
     1*(V2-1.0)+1.0)/(V2*S2)))                                                          36
      RETURN                                                                            37
```

FIGURE 13 - page iv

```
                                                                                Statement Numbers
50   FF=X(8)-V1*(AL1**((S1-1.0)/(V1*S1)))*(X(3)**(-1.0/S1))*(X(6)**((S1              * 38
    1*(V1-1.0)+1.0)/(V1*S1)))*X(5)                                                     39
     RETURN                                                                            40
C    INVESTMENT                                                                      * 41
55   FF=X(10)-(1.0/X(5))*(X(9)*C-XM)                                                 * 42
     RETURN                                                                            43
C    FULL EMPLOYMENT                                                                 * 44
60   FF=C-ABS(X(1))-ABS(X(2))                                                        * 45
     RETURN                                                                            46
65   FF=XL-ABS(X(3))-ABS(X(4))                                                       * 47
     RETURN                                                                            48
70   FF=X(6)-X(11)-X(12)-X(10)                                                       * 49
     RETURN                                                                            50
```

FIGURE 13- page v

```
                                                                          Statement Numbers
   SUBROUTINE PRINT (X,FFMAX,N,ITER)                                              1
   DIMENSION X(N)                                                                 2
   WRITE (6,5) ITER,FFMAX,X                                                       3
   RETURN                                                                         4
C                                                                                 5
5  FORMAT (14H ITERATION = ,I15/14H MAX VAL FF = .E15.6/14X,1HX/(28X,             6
  1E15.6))                                                                        7
   END                                                                            8
```

14-equation model are starred. Besides the function statements and the data relating to exogenous parameters and the initial guess, there are only five problem-specific Fortran statements (numbered 2, 48, 52, 2, and 7 in the right-hand column).

The ordering of the equations in function FF is important (see p. 48) and in Figure 13 is controlled by the following statement number 7:

GO TO (60, 65, 70, 25, 30, 15, 20, 5, 10, 35, 40, 45, 50, 55), J.

The equations will be treated by the algorithm in the order that they are listed in this GO TO statement. The linear equations 60, 65, and 70 thus appear first, the quadratic equations second, etc. Terms of the form X(i)** FP, where FP is a real noninteger number, are considered to be of the same degree as the number of the last term in the series expansion of $e^{FP(\log X(i))}$ which adds a significant digit on the particular computer being used (see footnote 19).

Where NONLIN or ZSYSTM is available, implementing Brown's method requires a statement of the model in implicit function form plus a small Fortran deck. Only five statement modifications are required to use the deck on other models.

2. Implementing a Dampened Jacobi Method.--To prepare the model for a Jacobi algorithm, the equations must be normalized, with each endogenous variable appearing exactly once on the left side of an equation. The following equations of the Kelly-Williamson-Cheetam model were modified to obtain the normalized version of the model.

3) $L_1 = \left[(P/w) \cdot V_1 \cdot Q_1^{\left[(\sigma_1(V_1-1)+1.0)/V_1\sigma_1\right]} A_1^{(\sigma_1-1)/V_1\sigma_1}\right]^{\sigma_1}$

5) $K_1 = \left[(P/r) \cdot V_1 \cdot Q_1^{\left[(\sigma_1(V-1)+1)/V_1\sigma_1\right]} A_1^{(\sigma_1-1)/V_1\sigma_1}\right]^{\sigma_1}$

11) $P = (r \cdot C - XM)/I$

12) $K_2 = C - K_1$

13) $L_2 = L - L_1$

14) $I = Q_1 - D_{11} - D_{12}$.

The equations must also be written in terms of the vector elements corresponding to the endogenous variables (see p. 59). In Jacobi's method (or any dampened method) dual vectors must be maintained, one for the previous iterate and one for the undampened figures used in generating the new iterate. Call the vector containing the previous iterate XL and the vector containing the undampened new iterates X. Then, in each equation of the dampened Jacobi algorithm, the endogenous variables on the right-hand side are replaced by the corresponding elements of XL. The left-hand sides are written as members of X. The market-balance equations, for instance, would be written as follows:

```
12  X(2) = C-XL (1)
13  X(4) = L - XL(3)
14  X(10) = XL(6) - XL(11) = XL(12)
```

A statement of the 14-equation model written in this form is included in the program presented in Figure 14.

Figure 14 is a copy of a simple algorithm written by the author to implement the dampened Jacobi method. This program calculates new undampened

FIGURE 14- page 1

```
                                                                    Statement Numbers
      PROGRAM SOR (OUTPUT,TAPE6=OUTPUT)                                     1
      DIMENSION X(100), XL(100)                                             2
C     CONTROL OF MAXIMUM NUMBER OF ITERATIONS                               3
      IMAX=5000                                                             4
C     RELAXATION PARAMETER                                                  5
      RLXP=0.25                                                             6
C     CRITICAL VALUE FOR TEST OF CONVERGENCE                                7
      CRIT=0.00000001                                                       8
C     PARAMETER VALUES                                                      9
      N=14                                                                 10
      C=30.0                                                             * 11
C     TL EQUIVALENT TO L, TOTAL LABOR                                    * 12
      TL=100.0                                                           * 13
      G=0.648                                                            * 14
C     S1 EQUIVALENT TO SIGMA 1                                           * 15
      S1=0.5                                                             * 16
      R1=(1.0-S1)/S1                                                     * 17
C     S2 EQUIVALENT TO SIGMA 2                                           * 18
      S2=1.5                                                             * 19
      R2=(1.0-S2)/S2                                                     * 20
      V1=1.0                                                             * 21
      V2=1.0                                                             * 22
      GT=0.0                                                             * 23
      XM=0.0                                                             * 24
C     AL1 EQUIVALENT TO A1                                               * 25
      AL1=0.64                                                           * 26
C     AL2 EQUIVALENT TO A2                                               * 27
      AL2=0.35                                                           * 28
      B11=0.8                                                            * 29
      B21=0.2                                                            * 30
      B12=0.5                                                            * 31
      B22=0.5                                                              32
C     INITIAL VALUES OF X(I)                                               33
C     INITIAL VALUES OF X(I), I LESS THAN OR EQUAL TO N                    34
      X(1)=18.174                                                        * 35
      X(2)=12.0                                                          * 36
      X(3)=42.7                                                          * 37
      X(4)=60.0                                                          * 38
      X(5)=11.0                                                          * 39
```

FIGURE 14- page ii

```
                                                                      Statement Numbers
      X(6)-10.0                                                            * 40
      X(7)=95.0                                                            * 41
      X(8)=0.9                                                             * 42
      X(9)=2.7                                                             * 43
      X(10)=2.0                                                            * 44
      X(11)=2.0                                                            * 45
      X(12)=2.0                                                            * 46
      X(13)=21.0                                                           * 47
      X(14)=45.0                                                           * 48
C     INITIAL VALUES OF X(I), I BETWEEN N AND 100                            49
      NP1=N+1                                                                50
      DO 5 JJ=NP1,100                                                        51
5       X(JJ)=0.0                                                            52
C     END OF INITIAL VALUES, SET XL=X                                        53
      DO 10 KR=1,100                                                         54
10      XL(KR)=X(KR)                                                         55
C     LIST OF NORMALIZED FNS, X(I)=F(XL(I))                                  56
      T1=CPTIME(1.0)                                                         57
      PRINT 55, T1                                                           58
15    CONTINUE                                                               59
C     PRODUCTION RELATIONSHIPS                                             * 60
      X(6)=AL1*(((XL(1)**(-R1))+(XL(3)**(-R1)))**((-V1)/R1))               * 61
      X(7)=AL2*(((XL(2)**(-R2))+(XL(4)**(-R2)))**((-V2)/R2))               * 62
C     FACTOR RETURNS                                                       * 63
      X(8)=(AL2**((S2-1.0)/(V2*S2)))*(XL(7)**((S2*(V2-1.0)+1.0)/(V2*S2))   * 64
     1)*(XL(4)**(-1.0/S2))*V2                                                65
      X(9)=(AL2**((S2-1.)/(V2*S2)))*(XL(7)**((S2*(V2-1.0)+1.0)/(V2*S2)))   * 66
     1*(XL(2)**(-1.0/S2))*V2                                                 67
      X(1)=((XL(5)*V1*(XL(6)**((S1*(V1-1.0)+1.0)/(V1*S1)))*(AL1**((S1-1.   * 68
     10)/(V1*S1)))*(1.0/XL(9)))**(S1))                                       69
      X(3)=((XL(5)*V1*(XL(6)**((S1*(V1-1.0)+1.0)/(V1*S1)))*(AL1**((S1-1.   * 70
     10)/(V1*S1)))*(1./XL(8)))**(S1))                                        71
C     INVESTMENT                                                             72
      X(5)=(XL(9)*C-XM)*(1.0/XL(10))                                       * 73
C     COMMODITY DEMAND                                                     * 74
      X(13)=XL(3)*(G+B21*(XL(8)-G))                                        * 75
      X(14)=XL(4)*(G+B22*(XL(8)-G))                                        * 76
      X(11)=XL(3)*B11*(XL(8)-G)/XL(5)                                      * 77
      X(12)=XL(4)*B12*(XL(8)-G)/XL(5)                                      * 78
C     FULL EMPLOYMENT                                                      * 79
```

FIGURE 1 4 - page iii

```
                                                              Statement Numbers
      X(2)=ABS(C-ABS(XL(1)))                                        *  80
      X(4)=ABS(TL-ABS(XL(3)))                                       *  81
      X(10)=ABS(XL(6)-ABS(XL(11))-ABS(XL(12)))                      *  82
C     END OF MODEL, END OF ONE ITERATION                               83
C     CHECK FOR CONVERGENCE                                            84
      DO 40 KC=1,N                                                     85
        IF (ABS(X(KC)-XL(KC))-CRIT) 40,40,20                           86
C     CHECK TO SEE MAXIMUM NUMBER OF ITERATIONS NOT EXCEEDED           87
20      IF (IT-IMAX) 30,45,25                                          88
25      STOP                                                           89
C     AUGMENT IT                                                       90
30      CONTINUE                                                       91
        IT=IT+1                                                        92
C     UPDATE XL                                                        93
        DO 35 I=1,N                                                    94
          XL(I)=RLXP*X(I)+(1.0-RLXP)*XL(I)                             95
35        CONTINUE                                                     96
C     RETURN TO BEGINNING OF MODEL AND PERFORM ANOTHER ITERATION       97
        GO TO 15                                                       98
40      CONTINUE                                                       99
      T2=CPTIME(1.0)                                                  100
      PRINT 60, T2                                                    101
C     CONVERGENCE ACHIEVED, PRINT SOLUTION                            102
      WRITE (6,65) IT                                                 103
      PRINT 70, ((I,X(I)),I=1,N)                                      104
      GO TO 50                                                        105
C     IF MAX ITERATIONS USED, PRINT LAST ITERATE                      106
45    CONTINUE                                                        107
      T3=CPTIME(1.0)                                                  108
      WRITE (6,75) T3                                                 109
      WRITE (6,80)                                                    110
      WRITE (6,85) ((I,X(I)),I=1,N)                                   111
50    CONTINUE                                                        112
      STOP                                                            113
C                                                                     114
55    FORMAT (30X,*T1=*,3X,F10.5)                                     115
60    FORMAT (30X,*T2=*,3X,F10.5)                                     116
65    FORMAT (20X,*ITERATIONS*,* USED *,*WERE *,I6)                   117
70    FORMAT (23X,*I=*,I3,2X,*X(I)= *,F21.14)                         118
75    FORMAT (30X,*T3=*,3X,F10.5)                                     119
80    FORMAT (10X,*MAX ITERATIONS USED  LAST X WAS*,/)                120
85    FORMAT (15X,*I=*,I3,2X,*MOST RECENT X(I)= *,F21.14)             121
      END                                                             122
```

iterates from the previous iterate, takes a weighted average, and iterates again until convergence is achieved or the maximum number of iterations equaled. Once again the cards which cannot be dispensed with but which must be modified for each specific problem are underlined (card number 10). Cards pertaining only to the 14-equation model are starred.

Programs for using Brown's method or the dampened Jacobi method are easily adapted for use on other models if the user can supply an appropriate Fortran statement of the model to be solved. Other iterative methods, such as Newton's or the Gauss-Seidel, are available at most computer facilities or can be written by the user.

APPENDIX B: The Relationship Between the Gauss-Seidel Method and the Cobweb Model, A More Mathematical Treatment

The cobweb model, originally worked out by Ezekial [7] to explain agricultural price movements, is a dynamic two-equation model well-known to economists. The cobweb model can be written in the form of a two-equation

Gauss-Seidel method, and the conditions under which each will converge to a fixed point can be shown to be equivalent. Demonstrating this result will provide an illustration of iterative methods and contraction mappings in terms more familiar to economists.

Consider the following model:

demand $\quad q^{(t)} = 12 - 3p^{(t)}$

supply $\quad q^{(t)} = 2 + 2p^{(t-1)}$,

where q indicates quantity of some commodity, p its price, and superscript t indicates the time period. The model is dynamic because it relates current year supply, and indirectly current year price, to price in the previous year. The model thus explains year-to-year changes in p and q. Only if $p = 2$ and $q = 6$ does the model show no change in either price or quantity. Thus, the point $(p,q) = (2,6)$ is an "equilibrium", or fixed point, of the cobweb model.

The cobweb version of the model given above would be written with the demand equation normalized on price, or

demand $\quad p^{(t)} = (q^{(t)} - 12)/-3$

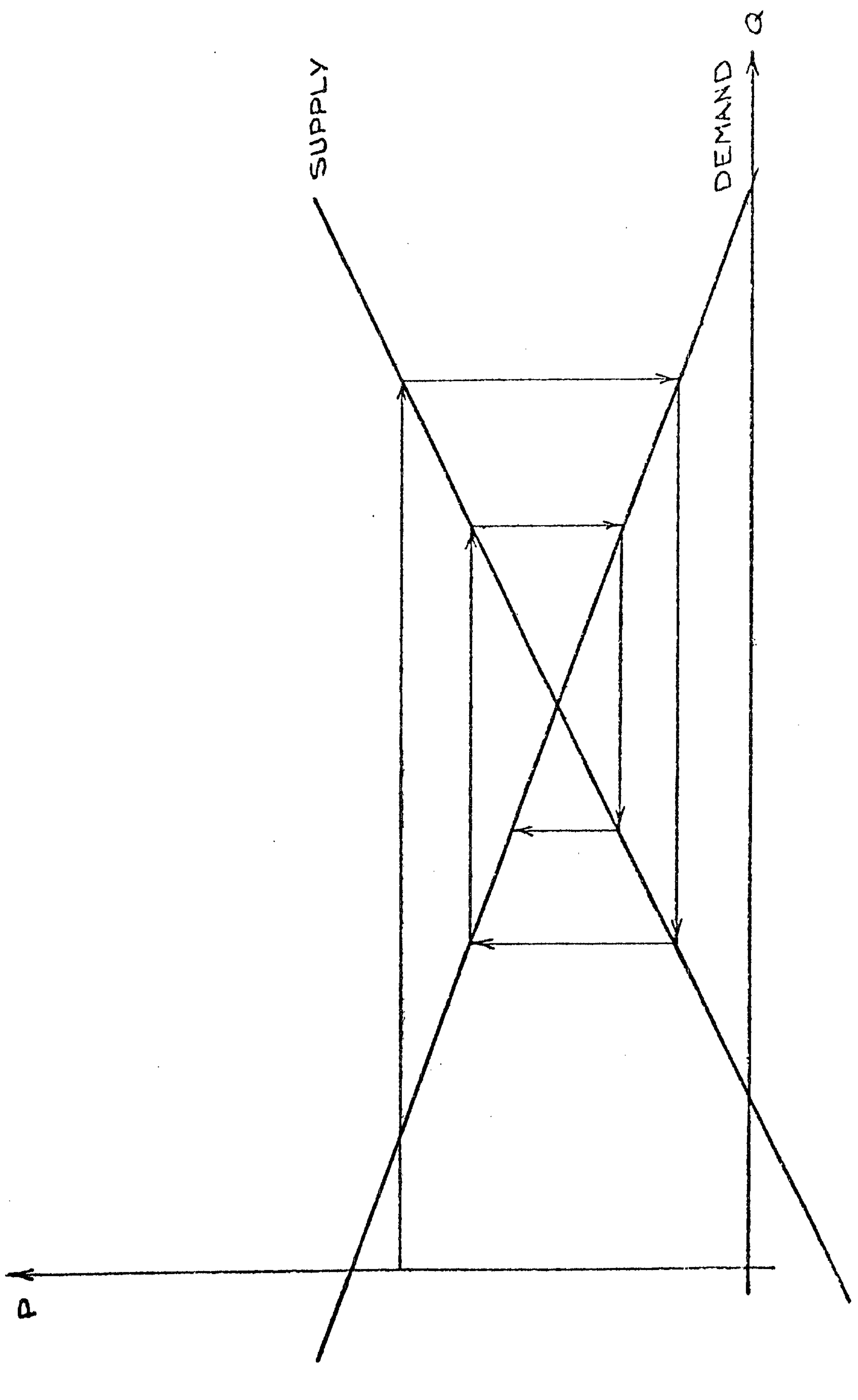
supply $\quad q^{(t)} = 2 + 2p^{(t-1)}$.

This form reflects the pattern of causality implied by the cobweb model. Previous year price determines current supply, which in turn determines current price. This form also conforms to the Gauss-Seidel method since each variable appears exactly once on the left-hand side of an equation. It is therefore possible to show the equivalence of the convergence criteria of the cobweb model and the Gauss-Seidel method.

The main result on convergence of the cobweb model is the cobweb theorem. This states that the sequence of price-quantity points of a cobweb model will converge to the stable or fixed point determined by the intersection of the supply and demand curve when the slope of the demand curve is of greater magnitude than the slope of the supply curve[26/] in a region containing some starting point. Since the slope of the demand curve is -3 in the above example and the slope of the supply curve is 2, the cobweb theorem indicates that the model would converge. Figure 15 illustrates this result. Note that the clockwise orientation of the "cobweb" is an implication of its pattern of causality. The path traced out moves horizontally towards the supply curve as previous price determines quantity and vertically towards the demand curve as current quantity determines current price.

The contraction mapping theorem and theorems 1 and 2 of Part I imply that sufficient conditions for the Gauss-Seidel method to converge are that the Jacobian of the iterative function have a spectral radius whose absolute value is less than 1 in a region containing the solution, and that this region be convex and be mapped into itself by the iterative function. These latter conditions are trivial in the linear case because the Jacobian is constant at all points. Thus, for the two equation model presented above,

FIGURE 15



it suffices to show that the iterative function has a Jacobian whose norm is less than 1. More general notation will then be employed to expand the analysis to higher dimensions.

In the general linear supply and demand model we have

supply $\quad q = r + sp$

demand $\quad q = t + vp$

The AX = B form of the system[27/] is

$$\begin{pmatrix} 1 & -s \\ 1 & -v \end{pmatrix} \begin{pmatrix} q \\ p \end{pmatrix} = \begin{pmatrix} r \\ t \end{pmatrix}.$$

Transforming this system to the form of the Gauss-Seidel method yields

$$q^{(k+1)} = r + sp^{(k)}$$

$$p^{(k+1)} = (t - q^{(k+1)})/-v,$$

or, in matrix form,

$$\begin{pmatrix} q^{(k+1)} \\ p^{(k+1)} \end{pmatrix} = \begin{pmatrix} r \\ \frac{t}{-v} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ \frac{1}{v} & 0 \end{pmatrix} \begin{pmatrix} q^{(k+1)} \\ p^{(k+1)} \end{pmatrix} - \begin{pmatrix} 0 & -s \\ 0 & 0 \end{pmatrix} \begin{pmatrix} q^{(k)} \\ p^{(k)} \end{pmatrix}.$$

Thus,

$$\left[\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ \frac{1}{v} & 0 \end{pmatrix}\right] \begin{pmatrix} q^{(k+1)} \\ p^{(k+1)} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -\frac{1}{v} & 1 \end{pmatrix} \begin{pmatrix} q^{(k+1)} \\ p^{(k+1)} \end{pmatrix} = \begin{pmatrix} 0 & s \\ 0 & 0 \end{pmatrix} \begin{pmatrix} q^{(k)} \\ p^{(k)} \end{pmatrix} + \begin{pmatrix} r \\ t/-v \end{pmatrix};$$

$$\begin{pmatrix} q^{(k+1)} \\ p^{(k+1)} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -\frac{1}{v} & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0 & s \\ 0 & 0 \end{pmatrix} \begin{pmatrix} q^{(k)} \\ p^{(k)} \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ -\frac{1}{v} & 1 \end{pmatrix}^{-1} \begin{pmatrix} r \\ t/-v \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ \frac{1}{v} & 1 \end{pmatrix} \begin{pmatrix} 0 & s \\ 0 & 0 \end{pmatrix} \begin{pmatrix} q^{(k)} \\ p^{(k)} \end{pmatrix} + N$$

$$= \begin{pmatrix} 0 & s \\ 0 & s/v \end{pmatrix} \begin{pmatrix} q^{(k)} \\ p^{(k)} \end{pmatrix} + N,$$

where $N = \begin{pmatrix} 1 & 0 \\ -\frac{1}{v} & 1 \end{pmatrix}^{-1} \begin{pmatrix} r \\ \frac{t}{-v} \end{pmatrix}$ is a vector of constants. The Jacobian of this linear iterative method is the matrix $\begin{pmatrix} 0 & s \\ 0 & \frac{s}{v} \end{pmatrix}$, which shall be denoted M. The eigenvalues of M are given by the formula $-\lambda_1 \left(\frac{s}{v} - \lambda_2\right) = 0.$[28] Thus, $\lambda_1 = 0$, $\lambda_2 = \frac{s}{v}$, and if $|v| > |s|$, $\rho(M) = |\lambda_2| < 1$ and the Gauss-Seidel iterative method will converge. The criterion $|v| > |s|$ is exactly the same as in the cobweb theorem. As indicated above, in the current model $|v| = 3 > 2 = |s|$, so convergence is confirmed by the theorems of both economics and numerical analysis. This shows that the cobweb model can be thought of as a mapping whose contractiveness depends on the relative slopes of the demand and supply curves. It also shows that economists are familiar with the problems of convergence of an iterative method even if by another name. The cobweb model is a Gauss-Seidel method adapted to a two-equation model and embodying a particular pattern of causality in its normalization.

The above analysis can now be extended to the general linear case AX = B. Taking the Gauss-Seidel iterative equations for the linear case (p. 16) and dividing through by a_{ii} yields

$$x_i^{(k+1)} = \frac{bi}{aii} - \sum_{j \le (i-1)} \frac{aij}{aii} x_j^{(k+1)} - \sum_{j \ge (i+1)} \frac{aij}{aii} x_j^{(k)}.$$

The first term is the ith element of $D^{-1}B$, where D is a diagonal matrix of the diagonal elements of A, $D = \text{diag}(a_{ii})$. The second term is the ith element of $-LX^{(k+1)}$, where L is a lower triangular matrix of elements of A, each divided by their corresponding diagonal element. That is, $L = (\ell_{ij})$ where

$$\ell_{ij} = \begin{Bmatrix} 0, & \text{if } j \geq i \\ \frac{aij}{aii}, & \text{if } j < i \end{Bmatrix}.$$

The third term is the ith element of $-UX^{(k)}$, where U is defined as L but on the upper triangular elements of A. $U = (u_{ij})$, where

$$u_{ij} = \begin{Bmatrix} 0, & \text{if } j \leq i \\ \frac{aij}{aii}, & \text{if } j > i \end{Bmatrix}$$

Thus

$$X^{(k+1)} = -LX^{(k+1)} - UX^{(k)} + D^{-1}B.$$

Rearranging this yields

$$X^{(k+1)} + LX^{(k+1)} = -UX^{(k)} + D^{-1}B,$$

or

$(I + L)X^{(k+1)} = -UX^{(k)} + D^{-1}B$. Multiplying through by $(I + L)^{-1}$ reveals the Gauss-Seidel iterative function,

$$X^{(k+1)} = -(I + L)^{-1}UX^{(k)} + (I + L)^{-1}D^{-1}B = MX^{(k)} + N,$$

where $M = -(I+L)^{-1}U$ and $N = (I+L)^{-1}D^{-1}B$.

Since the last term is a constant, the Jacobian of the function is M, and the Gauss-Seidel method will coverge if $\rho(M) < 1$, as was the case in the model as normalized above.

If instead the model had been normalized with a price-dependent supply equation and quantity-dependent demand equation, the results would have been exactly opposite, as shown below. This alternative model,

demand $\quad q = 12 - 3p$

supply $\quad p = (2-q)/2$

can be written in the form AX = B as

$$\begin{pmatrix} 1 & 3 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} q \\ p \end{pmatrix} = \begin{pmatrix} 12 \\ 2 \end{pmatrix}.$$

The Jacobian of the iterative function for this model is given by the matrix M, defined on page 77 .

$$M = -\begin{pmatrix} 1 & 0 \\ \frac{1}{2} & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 3 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ -\frac{1}{2} & -1 \end{pmatrix} \begin{pmatrix} 0 & 3 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -3 \\ 0 & -\frac{3}{2} \end{pmatrix}.$$

Then eigenvalues of M satisfy

$$(-\lambda_1)(-\tfrac{3}{2} - \lambda_2) = 0, \text{ or } \lambda_1 = 0, \ \lambda_2 = \tfrac{3}{2}.$$

Thus, $\rho(M) = \frac{3}{2} > 1$, and the Gauss-Seidel iteration based on this normalization diverges. The model in effect becomes a cobweb in reverse, tracing a counter-clockwise and divergent path as shown in Figure 6.

APPENDIX C

Van Der Geissen's Method for Ordering Equations Prior to Gauss-Seidel Iteration

A method for ordering the equations of an economic model in a recursive pattern amenable to solution via Gauss-Seidel iteration has been reported by A. A. van der Giessen [22]. A simple computer program for implementing the technique is presented in this appendix and applied to the 14-equation Kelley-Williamson model explained in Appendix A (also see [15]).

To use the technique, one must express the model in the normalized form required for Gauss-Seidel iteration (i.e., each of the n endogenous variables must be isolated exactly once on the left-hand side of an equation). The user then supplies a set of zero-one variables indicating the structure of the model, and the algorithm automatically orders the equations in a more recursive pattern. Increased recursiveness reduces the number of endogenous variables whose initial estimate must be supplied by the user and enhances the chances of achieving convergence via Gauss-Seidel iteration.

The pattern of interdependence among the endogenous variables is represented by the zero-one elements of the "interdependency matrix" supplied by the user. This matrix has n rows and n columns, one for each endogenous variable in the model to be solved. A 1 in the ij^{th} position indicates that the i^{th} variable depends on the j^{th} variable; a 0 indicates that the i^{th} variable does not depend on the j^{th} variable.29/ Figure 16 gives the interdependency matrix for the normalized version of the Kelley-Williamson model presented in Appendix A.

To implement the ordering routine, three additional columns are appended to the interdependency matrix. The $n+1^{st}$ column is called the "auxiliary" column. Its elements are equal to the row sums of the first n columns. Thus, the i^{th} row of the auxiliary column lists the total number of endogenous variable appearing in the equation of the i^{th} variable. The $n+2^{nd}$ and $n+3^{rd}$ columns are for accounting purposes. The n+3rd column lists

FIGURE 16

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Auxiliary Column	Variables Whose Initial Guess Must Be Given	Order in Which Equations Are to Be Solved
1					1	1			1						3		
2	1														1		
3					1	1		1							3		
4	1														1		
5									1	1					2		
6	1		1												2		
7		1		1											2		
8				1			1								2		
9		1					1								2		
10						1				1	1				3		
11			1		1			1							3		
12				1	1			1							3		
13			1					1							2		
14				1				1							2		

Figure 17.

Variables Whose Initial Guess Must be Given	Order in Which Equations Are to Be Solved
1	2
3	4
11	7
12	8
	9
	14
	6
	13
	10
	5
	1
	3
	11
	12

the equations in the order in which they should be solved. For example, if 2 appears in the first row of the n+3rd column, the equation for the 2nd variable must be solved first in implementing the Gauss-Seidel method. The n+2nd column lists the variables for which an initial guess must be supplied by the user. Figure 17 shows the results of applying the algorithm to the 14-equation model.

From the initially supplied interdependency matrix, the method proceeds as follows:

1. Calculate the auxiliary column (for variables not already listed in in the n+3rd column).
2. Check for zeroes in the auxiliary column. If a zero appears in the i^{th} column, the i^{th} variable depends on no remaining endogenous variables and is therefore predetermined. Accordingly, the i^{th} column is deleted (set equal to zero). To indicate that the i^{th} variable is next in the order of equations to be solved, the value i is placed in the highest unoccupied cell in column n+3. A large number M $(M > n)$ is placed in the i^{th} cell of the auxiliary column. The method returns to step 1.
3. After all zeroes are cleared from the auxiliary column, check for 1's on the main diagonal of the interdependency matrix (i.e., the ii^{th} cells). Initially there should be no 1's on the main diagonal if the user has properly normalized the model. However, 1's may be introduced into the ii^{th} cell of the main diagonal as the method proceeds. In this case they indicate that the i^{th} variable can no longer be expressed as a function of the remaining endogenous variables and must therefore be listed in column n+2 among the variables for which an initial guess must be given by the user. If so, the i^{th} column is deleted. The method then returns to step 1.

4. After all 1's are cleared from the main diagonal, an index variable r is set equal to one.
5. Check the auxiliary column for an r. If the ith cell of the auxiliary column is r, the ith variable is a function of r endogenous variables. If r=1, the ith variable is a function of just one other variable j and can be replaced by substituting x_j for x_i wherever x_i appears. This is accomplished by "adding" the ith column to the jth column, where "addition" is defined as $\left\{\begin{matrix} 0+0=0 \\ 0+1=1+0=1 \\ 1+1=1 \end{matrix}\right\}$. In this manner variables dependent upon X_i are made dependent upon X_j as well. If $r > 1$, the procedure is the same except that 1's from column i are "added" to the r other columns containing a 1.

   After the first r has been processed in this manner, go to step 7.

   If no r is found, go to step 6 (for variables not already listed in the n+3rd column).
6. If no r appears in the auxiliary column, increase r by one and repeat step 5.
7. Recalculate the auxiliary matrix.
8. Check for 1's on the main diagonal. If a 1 is found, proceed as indicated in step 3 and return to step 1. If a 1 is not found, go to step 9.
9. Check the remainder of the auxiliary column for an r. If found, return to step 5 and proceed as indicated there. Otherwise increase r by one and then return to step 5.
10. Terminate the method listed in Steps 1-9 when all equations have been ordered, that is when column n+3 is full.

A flow-chart of this method is shown in Figure 18 and a program for implementing the method on the 14-equation model listed in Figure 19.

FIGURE 18

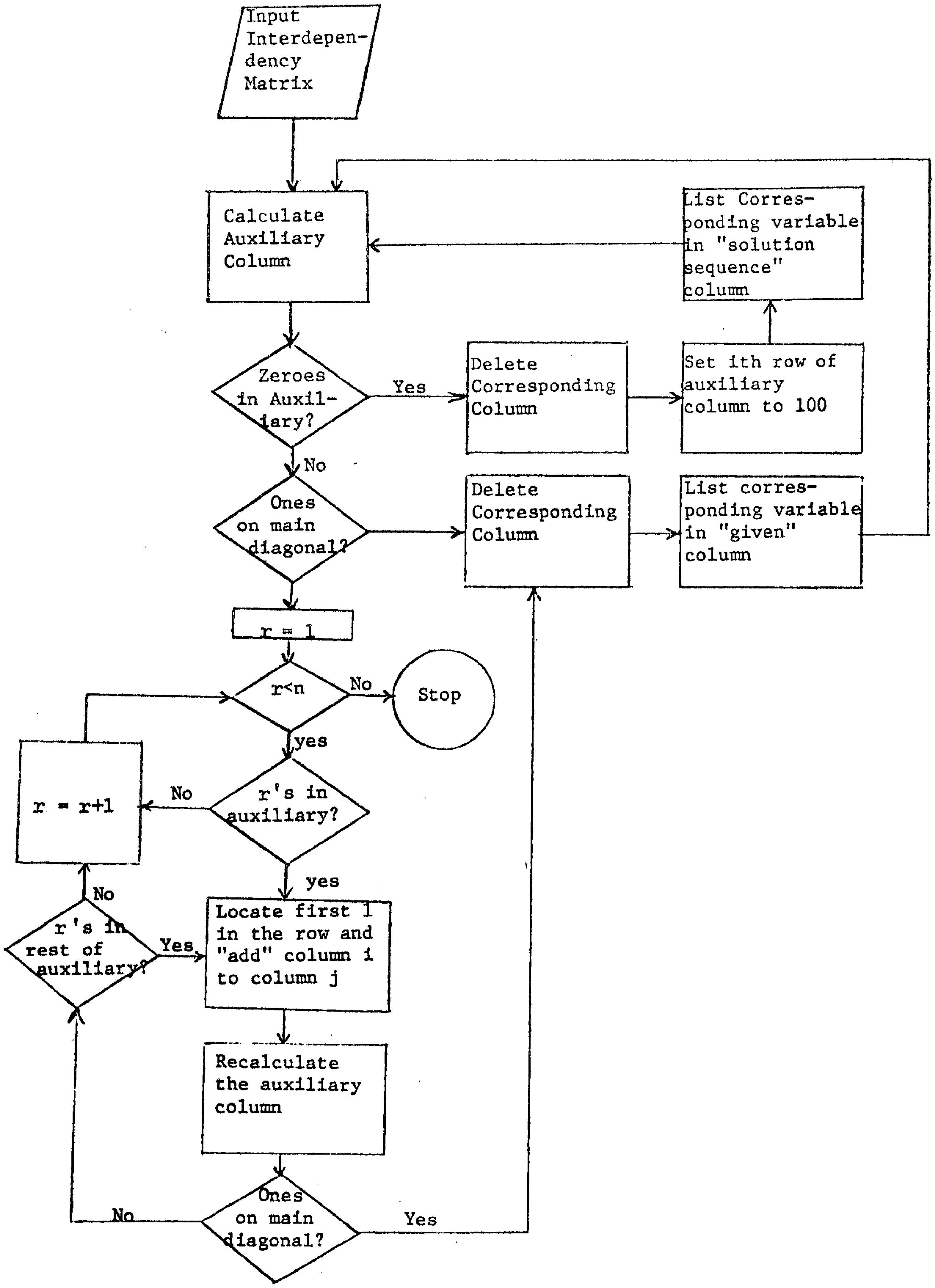

Input Interdependency Matrix
Calculate Auxiliary Column
List Corresponding variable in "solution sequence" column
Zeroes in Auxiliary?
Yes
Delete Corresponding Column
Set ith row of auxiliary column to 100
No
Ones on main diagonal?
Delete Corresponding Column
List corresponding variable in "given" column
r = 1
r<n
No
Stop
yes
r's in auxiliary?
No
r = r+1
yes
No
r's in rest of auxiliary?
Yes
Locate first 1 in the row and "add" column i to column j
Recalculate the auxiliary column
Ones on main diagonal?
No
Yes

FIGURE 19 page 1

```
                                                                Statement Numbers
      PROGRAM SIMSEQ (INPUT,OUTPUT)                                     1
      DIMENSION M(80,83)                                                2
      DIMENSION A(3,2)                                                  3
C     INITIALIZE VARIABLES                                              4
      N=14                                                              5
      IY=101                                                            6
      IX=1                                                              7
      N3=N+3                                                            8
      NGIV=1                                                            9
      NSEQ=1                                                           10
C     READ FORMATS FOR DATA AND COLUMN HEADS                           11
C     INPUT ZERO-ONE COEFFICIENTS OF INTERDEPENDENCY                   12
      DO 5 I=1,N                                                       13
5       READ 145, (M(I,J),J=1,80)                                      14
      READ 150, ((A(IA,J),J=1,2),IA=1,3)                               15
C     RECALCULATE AUXILIARY COLUMN                                     16
C     CHECK FOR 100#S IN AUXILIARY COLUMN                              17
10    DO 30 K=1,N                                                      18
        IF (M(K,N+1)-100) 20,30,15                                     19
15      STOP 400                                                       20
C     CALCULATE NEW ROW SUMS FOR AUXILIARY COLUMN                      21
20      M(K,N+1)=0                                                     22
        DO 25 J=1,N                                                    23
25        M(K,N+1)=M(K,N+1)+M(K,J)                                     24
30      CONTINUE                                                       25
C     CHECK FOR ZEROS IN AUXILIARY COLUMN                              26
      DO 50 K=1,N                                                      27
        IF (M(K,N+1)-0) 35,40,50                                       28
35      STOP 401                                                       29
40      CONTINUE                                                       30
C     PUT ZEROS IN COL K AND PLACE VAR K IN SEQUENCE                   31
        DO 45 L=1,N                                                    32
45        M(L,K)=0                                                     33
C     SET CORRESPONDING ELEMENT OF AUXILIARY COLUMN TO 100             34
        M(K,N+1)=100                                                   35
C     LIST CORRESPONDING VARIABLE IN SOLUTION SEQUENCE COLUMN          36
        M(NSEQ,N+3)=K                                                  37
C     UPDATE SOLUTION SEQUENCE PARAMETER                               38
        NSEQ=NSEQ+1                                                    39
```

FIGURE 19 - page ii

```
                                                                        Statement Numbers
     GO TO 10                                                                  40
50   CONTINUE                                                                  41
C   CHECK FOR ONES ON MAIN DIAGONAL                                            42
    DO 70 J=1,N                                                                43
     IF (M(J,J)-1) 70,60,55                                                    44
55   STOP 402                                                                  45
C   DELETE CORRESPONDING COLUMN                                                46
60   DO 65 I=1,N                                                               47
65     M(I,J)=0                                                                48
C   LIST CORRESPONDING VARIABLE IN GIVEN COLUMN                                49
     M(NGIV,N+2)=J                                                             50
C   UPDATE GIVEN VARIABLE PARAMETER                                            51
     NGIV=NGIV+1                                                               52
     GO TO 10                                                                  53
70   CONTINUE                                                                  54
C   SEARCH AUX COL FOR LOWEST VALUES                                           55
    DO 130 NO=1,N                                                              56
C   CHECK FOR VARIABLE NO IN AUXILIARY COLUMN                                  57
C   VARIABLE NO CORRESPONDS TO VARIABLE K IN FLOW CHART                        58
     DO 125 K=1,N                                                              59
       IF (M(K,N+1)-NO) 125,75,125                                             60
C   CHECK FOR FIRST ONE IN CORRESPONDING ROW                                   61
75     DO 95 J=I,N                                                             62
         IF (M(K,J)-1) 95,85,80                                                63
80       STOP 404                                                              64
C   ≠ADD≠ COLUMN I TO COLUMN J                                                 65
85       DO 90 L=1,N                                                           66
90         M(L,J)=M(L,K)+M(L,J)-M(L,K)*M(L,J)                                  67
         GO TO 100                                                             68
95       CONTINUE                                                              69
100    CONTINUE                                                                70
C   RECALCULATE AUXILIARY COLUMN                                               71
       DO 115 KO=1,N                                                           72
         IF (M(KO,N+1)-100) 105,115,15                                         73
105      M(KO,N+1)=0                                                           74
         DO 110 J=1,N                                                          75
110        M(KO,N+1)=M(KO,N+1)+M(KO,J)                                         76
115      CONTINUE                                                              77
C   CHECK FOR ONES ON MAIN DIAGONAL                                            78
       DO 120 J=1,N                                                            79
         IF (M(J,J)-1) 120,60,55                                               80
```

FIGURE 19 - page iii

```
                                                          Statement Numbers
120        CONTINUE                                              81
125      CONTINUE                                                82
130    CONTINUE                                                  83
    DO 135 JA=1,2                                                84
135   PRINT 155, (A(I,JA),I=1,3)                                 85
    DO 140 I=1,N                                                 86
140   PRINT 160, I,(M(I,J),J=N+1,N+3)                            87
    STOP                                                         88
C                                                                89
145 FORMAT (80I1)                                                90
150 FORMAT (6A10)                                                91
155 FORMAT (30X,A10,2X,A10,2X,A10)                               92
160 FORMAT (27X,I2,4X,I3,9X,I3,9X,I3                             93
    END                                                          94
```

FOOTNOTES

1/ Iterative methods can be more generally expressed as $X^{(k+1)} = G(X^{(k)}, X^{(k-1)}, \ldots, X^{(k-s)})$. For the methods discussed in this paper $s = 0$ and $X^{(k+1)} = G(X^{(k)})$. Many useful methods are of this form [see 17, Ch. 7; 4, Ch. 1].

2/ This is, if for any $\varepsilon > 0$, there is integer M such that for any $k > M$, $D[F(X^{(k)}) - F(X^{(*)})] < \varepsilon$.

3/ A fixed point of a function G(X) is a point X* where the value of the function is equal to its argument. That is, a point X* such that $X^* = G(X^*)$.

4/ Justification for this result is given by the mean value theorem of calculus, which states that between any two points on a continuous and differentiable segment of a function G(x) there is a third point where the slope of the function equals the average rate of change of the function between the first two points. That is, if G(x) is continuous and differentiable on the interval $[x^{(k)}, x^{(k-1)}]$, then

$$\frac{[G(x^{(k)}) - G(x^{(k-1)})]}{[x^{(k)} - x^{(k-1)}]} = G'(\bar{x}),$$

where $\bar{x}$ lies in the interval between $x^{(k)}$ and $x^{(k-1)}$. Rearranging terms and recalling that $[x^{(k+1)} - x^{(k)}] \equiv [G(x^{(k)}) - G(X^{(k-1)})]$, it can be seen that $[x^{(k+1)} - x^{(k)}] = [x^{(k)} - x^{(k-1)}]G'(\bar{x})$. If $G'(x)$ is less than one for all x in a neighborhood of x* containing $x^{(o)}$, then the absolute value of the differences between successive iterates grows progressively smaller and tends to zero. The iterates would form a Cauchy sequence converging to the limiting fixed point value x*.

5/ See [19, p. 265].

6/ These properties are derived from [4, p. 175].

7/ An eigenvalue of a matrix A, here denoted $\lambda(A)$, is a number such that $AX = [\lambda(A)]X$, for some vector X. See [13, Chapter 7].

8/ Functions with n-dimensional domains and m-dimensional ranges are also mappings but are not discussed here.

9/ Ortega and Rheinboldt [17, p. 69] show that if a mapping G from R^n to R^m has a well-defined Jacobian on a convex set $D' \subset D$, then for any X, $Y \in D'$,

$||G(Y) - G(X)|| \leq \sup_{0 \leq t \leq 1} ||G'(X+t(Y-X))|| \cdot ||X-Y||$. Since for $0 \leq t \leq 1$ $[X+t(Y-X)] = [(1-t)\ X+tY] \in D'$ by convexity, and since it is assumed above that $||G'(X)|| \leq \alpha < 1$ for all points in D', it follows that $||G(Y) - G(X)|| \leq \alpha\ ||X-Y||$. Q.E.D.

10/ Ortega and Rheinboldt [17, p. 43] show that $\rho(A) \leq ||A||$ for all norms, but that for any number $\varepsilon > 0$, there is some choice of norm such that $||A|| < \rho(A) + \varepsilon$ [17, p. 44]. If $\rho(A) < 1$, choose $0 < \varepsilon < 1 - \rho(A)$. The theorem follows.

11/ Only strictly true if the subset is convex.

12/ Larger than 100 x 100. Exact methods are also inappropriate if any equation is nonlinear.

13/ Note that by interchanging rows and columns of A it is always possible to write the system AX = B in such a way that $A_{ii} \neq 0$, i = 1, ..., n, as long as A is nonsingular.

14/ There are systems of equations where this normalization procedure is either impossible or exceedingly difficult. In such cases a secondary iterative method must be supplied to solve for each $x_i^{(k+1)}$ from a nonlinear implicit function of the form $f_i(x_1^{(k)}, \ldots, x_{i-1}^{(k)}, x_{i+1}^{(k)}, \ldots, x_n^{(k)} = 0$ in the Jacobi method or $f_i(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, x_{i+1}^{(k)}, \ldots, x_n^{(k)})$ in the Gauss-Seidel method. The authors feel that this situation will rarely arise in economic models since they are usually constructed with fairly simple functional forms.

15/ There are the following two additional patterns if the ordering of equations is considered:

3. (clockwise)

   Demand $p^{(t)} = (\frac{-1}{3})\ q^{(t-1)} + 4$

   Supply $q^{(t)} = 2 + 2\ p^{(t)}$

4. (Counter clockwise)

   Demand $q^{(t)} = 12 - 3\ p^{(t-1)}$

   Supply $p^{(t)} = (\frac{1}{2})\ q^{(t)} - 1$

In this 2-equation, linear model only the normalization, and not the ordering, of equations affects convergence. Ordering does affect Gauss-Seidel convergence in the general case.

16/ The Picard Method, the Secant Method and the Steffensen Method are some of the other important linearization methods. They are discussed in numerical methods texts such as [4] and [17].

17/ One justification for choosing $A = J$ is derived from the spectral radius convergence criterion. Let X^* be a solution of $F(X) = 0$. Then, in a neighborhood of X^*, convergence is assured if the Jacobian of $X^{(k+1)} = X^{(k)} + A^{-1} F(X^*)$ has spectral radius of magnitude less than one. At X^* this Jacobian can be written $[I - A^{-1} F'(X^*)]$. If $\lim_{k \to \infty} X(k) = X^*$, then $\rho J = \rho[I - A^{-1} F'(X^*)]$ will approach zero if $A = F'(X^{(k)})$.

18/ The approximation may also be thought of as a Taylor series expansion.

19/ Logarithmic, exponential, or trigonometric expressions are assigned a degree equal to the last term in their infinite series expansion which is significantly different from zero (determined by the machine tolerance of the computer on which the method is implemented.

20/ See [12, p. 71].

21/ It is possible, however, to introduce singularity into the calculation of a Jacobian if Brown's method is altered in an attempt to preclude selection of negative iterates. For instance, one of the author once altered Brown's method in such a way as to substitute the value 0.1 for any negative element of an iterate. This caused several rows of zeros to appear in the discretely estimated Jacobian matrix used in Brown's method.

22/ Chiefly Professor Kenneth M. Brown, Department of Computer Science, University of Minnesota, and David M. Gay, Research Associate, National Bureau of Economic Research, Cambridge, Mass.

23/ Researchers in the Department of Agricultural and Applied Economics of the University of Minnesota have used Brown's method to solve systems of equations characterizing the necessary first-order conditions for maximization of likelihood functions. Singularity of the Jacobian has been encountered in these problems.

24/ This is still a weighted average as before, since $X^{(k)} = [(1-\delta) + \delta]$ $X^{(k)}$ and $[X^{(k)} + J^{-1} (X^{(k)}) F(X^{(k)})]$ is the undampened Newton iterate.

25/ See [4; 17].

26/ "Slope" in this context refers to the derivative of quantity with respect to price in both the supply and demand curve. These slopes are most easily evaluated when both equations are written with quantity as the dependent variable. See also [19, p. 265].

27/ Normalization in the AX = b form is implied by the order of the equations. The first equation is normalized on the first variable, q; and the second equation is normalized on the second variable, p. If the system were written as

$$\begin{pmatrix} 1 & -v \\ 1 & -s \end{pmatrix} \begin{pmatrix} q \\ p \end{pmatrix} = \begin{pmatrix} t \\ r \end{pmatrix}$$

, the result on convergence of the system would be exactly the opposite of the result that follows above

28/ See [13, Chapter 7].

29/ The interdependence matrix will change if the numbering of the variables is altered. Such changes can affect the order in which equations will be listed for solution and also the variables that will be listed as requiring an initial guess. Van der Geissen claims, however, that the changes are slight and do not significantly affect the Gauss-Seidel solution of the model [22, pp. 47-48].

Bibliography

1. Charles R. Blitzer, Peter B. Clark and Lance Taylor, editors, Economy-Wide Models and Development Planning, published for the World Bank by Oxford University Press (1975), pp. 95-103.

2. Brown, Kenneth M. "Computer Oriented Algorithms for Solving Systems of Simultaneous Nonlinear Algebraic Equations," in Numerical Solutions of Systems of Nonlinear Algebraic Equations, ed. G. D. Byrne and C. A. Hall. New York: Academic Press, 1973, pp. 281-348.

3. ________, and J. E. Dennis, Jr. "On the Second Order Convergence of Brown's Derivative-Free Method for Solving Simultaneous Nonlinear Equations," Technical Report 71-7. New Haven, Conn.: Yale Univ. Dept. of Computer Science, 1971.

4. Dahlquist, Germund, and Ake Bjorck. Numerical Methods. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1974.

5. John H. Duloy and Roger D. Norton, "CHAC, A Programming Model of Mexican Agriculture" in Multi-Level Planning: Case Studies in Mexico, edited by Louis M. Goreux and Alan S. Manne.

6. ________, "Prices and Incomes in Linear Programming," AJAE, Vol. 57, No. 4 (Nov. 1975), pp. 591-600.

7. Ezekiel, Mordecai. "The Cobweb Theorem." Quarterly Journal of Econ., Vol. 53 (Feb. 1938).

8. Fisher, Michael E., and A. T. Fuller. "On the Stabilization of Matrices and the Convergence of Linear Iterative Processes." Proceedings of the Cambridge Philosophical Society, Vol. 54 (1958), pp. 417-425.

9. Gay, David. "On Modifying Singular Values to Solve Possibly Singular Systems of Nonlinear Equations." Working Paper No. 125 (preliminary, revised). Cambridge, Mass.: Computer Research Center for Economics and Management Science, National Bureau of Economic Research, Inc., June 1976.

10. ________. "Implementing Brown's Method." Publication CNA-109. Austin, Tex.: Univ. of Texas, Center for Numerical Analysis, Dec. 1975.

11. Martin Greenberger, Matthew A. Crewson and Brian L. Crissey, Models in the Policy Process, Russell Sage Foundation, 230 Park Avenue, N.Y., N.Y. (1976).

12. Heien, Dale, Jim Matthews, and Abner Womack. "A Methods Note on the Gauss-Seidel Algorithm for Solving Econometric Models." Agricultural Econ. Research, Vol. 25, No. 3 (July 1973), pp. 71-75.

13. Hadley, G. Linear Algebra. Reading, Mass.: Addison-Wesley, 1961.

14. Hurwicz, Leonid. "The Design of Mechanisms for Resource Allocation." Amer. Econ. Review, Vol. LXIII, No. 2 (May 1973).

15. Kelley, Allen C., Jeffrey G. Williamson, and Russell J. Cheetam. Dualistic Economic Development. Chicago: Univ. of Chicago Press, 1972.

16. Roger D. Norton and Pasquale L. Scandizzo, A Computable Class of General Equilibrium Models, Working paper No. 10, DRC, IBRD (March 1977).

17. Ortega, J. M. and W. C. Rheinboldt. Iterative Solution of Nonlinear Equations in Several Variables. New York: Academic Press, 1970.

18. Quirk, James, and Rubin Saposnik. Introduction to General Equilibrium Theory and Welfare Economics. New York: McGraw-Hill, 1968.

19. Samuelson, Paul. Foundations of Economic Analysis. Cambridge, Mass.: Harvard Univ. Press, 1966.

20. Harold T. Shapiro and Laslo Halabok, "Macro-Economic Model Building in Socialist and Non-Socialist Countries: A Comparative Study," International Economic Review, Vol. 17, No. 3 (Oct. 1976), pp. 529-565.

21. Todd, Richard, "Iterative Methods for Solving Simultaneous-Equation Models: Results from Theory and Application", Masters Plan B Project Paper, Department of Agricultural and Applied Economics, University of Minnesota, (December 1976).

22. Van der Geissen, A.A. "Solving Non-Linear Systems by Computer; A New Method." Statisca Neerlandica, Vol. 24 (Jan. 1970), pp. 41-50.