8010

GENERATING CAUSAL RELATIONS FROM MATHEMATICAL MODELS

by

TUNGA GüNGöR

B.S. in Computer Engineering, Boğaziçi University, 1987

Submitted to the Institute for Graduate Studies in Science and Engineering in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

Boğaziçi University

1989

GENERATING CAUSAL RELATIONS

FROM

MATHEMATICAL MODELS

APPROVED BY

Doç. Dr. Selahattin Kuru [illegible]

(Thesis Supervisor)

Prof. Dr. Yorgo Istefanopulos [illegible]

Doç. Dr. Oğuz Tosun [illegible]

DATE OF APPROVAL 2/10/1989

ACKNOWLEDGEMENTS

I would like to thank Doç. Dr. Selahattin Kuru for his great help and guidance as the supervisor of this thesis.

I also thank to Doç. Dr. Oğuz Tosun and Prof. Dr. Yorgo Istefanopulos for both their comments and serving on my thesis committee.

ABSTRACT

System analysis is an important topic for many scientific disciplines. By a system, we mean an orderly, interconnected arrangement of parts describing a phenomenon. The formal representation of systems is done through mathematical modeling. Thus the analysis of a system is performed by analysing the mathematical model of the system.

The mathematical model of a system contains implicit information about the system it describes - how the system behaves under various conditions, what are the relationships between the variables, how the cause-effect sequence of the variables be arranged, etc. Currently, the exposition of this information in order to understand the system truly is usually performed by humans. This thesis is a step towards the automation of this process and introduces for this purpose some techniques. These techniques are based on the use of some well-known mathematical tools: partial derivatives and total differentials of the closed form functions defining a system. Partial derivatives and total differentials are analysed to make the causal relations implied by the model explicit. The mathematical models addressed by our work are restricted to models of the form of algebraic equations.

We can classify the process of system analysis into two: Quantitative analysis and qualitative analysis. The techniques introduced in this work are mostly qualitative in nature, but they also take into account the quantitative information available in a model.

ÖZET

Sistem analizi pekçok bilim dalını ilgilendiren bir konudur. Bölümleri arasında düzenli bağlantılar bulunan herhangi bir olayı bir sistem olarak tarif edebiliriz. Bir sistem matematik modelleme yoluyla ifade edilir. Böylece bir sistemin analizi sisteme ait matematik modelin analizi ile yapılır.

Bir sistemin matematik modeli sistem hakkında önemli bilgiler içerir: değişik durumlarda sistem nasıl davranır, değişkenler arasındaki ilişkiler nelerdir, değişkenlere ait sebep-sonuç sırası ne şekilde düzenlenmiştir, vb. Günümüzde sistemi doğru olarak anlayabilmek için bu bilgilerin çıkarımı genellikle insanlar tarafından yapılmaktadır. Bu tez, bu işlemin otomasyonu yönünde bir adım atmayı amaçlamaktadır ve bu sebeple bazı teknikler ortaya koymaktadır. Bu teknikler yaygın olarak kullanılan birtakım matematik metotlara dayanmaktadır: bir sistemi tarifleyen kapalı fonksiyonların kısmi türevleri ve toplam türevleri. Kısmi ve toplam türevler modeldeki değişkenler arasındaki sebep-sonuç ilişkilerini ortaya çıkarmak amacıyla kullanılmaktadır. Bu çalışmada yer alan matematik modeller cebirsel denklemlerden oluşmaktadır.

Sistem analizi işlemini iki gruba ayırabiliriz: Sayısal analiz ve niteliksel analiz. Bu çalışmada ortaya atılan teknikler genel olarak nitelikseldir, ancak bir modelde bulunan sayısal bilgi de dikkate alınmaktadır.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	vi
TABLE OF CONTENTS	viii
LIST OF FIGURES	x
LIST OF TABLES	xi
I. INTRODUCTION	1
1.1. Mathematical Models and Qualitative Analysis	3
1.2. Summary of Our Approach	9
II. DEFINITIONS, NOTATION AND THE FUNDAMENTAL ASSUMPTION	11
III. SIGN ANALYSIS	24
3.1. Sign Tables	25
3.2. Sign Analysis of Total Differentials	39
3.3. Sign Analysis of Gains	60
IV. VALUE ANALYSIS	72
4.1. Gains	72
4.2. Value Analysis of Total Differentials	80
4.3. Elasticities	84

V. ILLUSTRATION OF THE TECHNIQUES 86

5.1. Assembling 88

5.2. Sign Analysis of Total Differentials 90

5.3. Sign Analysis of Gains 104

5.4. Gains 108

5.5. Elasticities 111

5.6. Value Analysis 112

VI. IMPLEMENTATION 115

6.1. Notes About the Program 115

6.2. Options 122

VII. CONCLUSION 129

APPENDIX A. NEWTON-RAPHSON METHOD 133

APPENDIX B. PROGRAM MESSAGES 135

APPENDIX C. SPECIFICATIONS 139

APPENDIX D. A SAMPLE RUN 141

APPENDIX E. PROGRAM LISTING 158

BIBLIOGRAPHY 201

LIST OF FIGURES

		Page
FIGURE 1.	Stages of quantitative analysis	2
FIGURE 2.	Stages of qualitative analysis	2
FIGURE 3.	A valve system	19
FIGURE 4.	The valve attached to a vessel	19
FIGURE 5.	A procedure for constructing the sign table of a function	35
FIGURE 6.	Number of rows, N, in the sign table of an n-parameter function	36
FIGURE 7.	Two sign tables for the join example	42
FIGURE 8.	Example of join operation	43
FIGURE 9.	Example system used in Section 5	87
FIGURE 10.	A pump system of two branches	89
FIGURE 11.	Reduced tables for the increase in Q_F	97
FIGURE 12.	Operator and function results in qualitative arithmetic	119
FIGURE 13.	Program main menu	123

LIST OF TABLES

		Page
TABLE 1.	Sign table of $F(z)=0$	29
TABLE 2.	Sign table of $F(z_1,z_2)=0$	29
TABLE 3.	Sign table of $F(z_1,z_2,z_3)=0$	29
TABLE 4.	Sign table of $F(z_1,z_2,z_3,z_4)=0$	30
TABLE 5.	Table of all sign combinations for $F(z_1,z_2)=0$	34
TABLE 6.	Sign table of $F=z_1^2-z_2=0$	45
TABLE 7.	Sign table of $F=z_2+z_3=0$	45
TABLE 8.	Pure sign table of $F=z_1^2-z_2=0$ for $[z_1]=+$	46
TABLE 9.	Result of Example 3.2.1 for $z_1>0$	47
TABLE 10.	Result of Example 3.2.2 for $z_1>0$	47
TABLE 11.	Pure sign table of $F=z_1^2-z_2=0$ for $[z_1]=-$	47
TABLE 12.	Result of Example 3.2.1 for $z_1<0$	47
TABLE 13.	Result of Example 3.2.2 for $z_1<0$	48
TABLE 14.	Result of Example 3.2.3 for $z_1<0$	48
TABLE 15.	Result of $\in$-join for $z_1>0$	49
TABLE 16.	Sign table of $F=z_1-z_2-z_3=0$	54
TABLE 17.	Sign table of $F=3z_1-2z_2-7=0$	54
TABLE 18.	Result of example in Proposition 3.3	54
TABLE 19.	Sign table of gains of $F=z_1^2-z_2=0$ wrt z_1	64
TABLE 20.	Sign table of gains of $F=z_2+z_3=0$ wrt z_1	64
TABLE 21.	Reduced form of Table 19	64
TABLE 22.	Result of sign analysis for gains	64
TABLE 23.	Sign table of gains of $F=y_1^2-x_1=0$ wrt y_1	70

TABLE 24. Sign table of gains of $F=x_1+x_2+y_2=0$ wrt y_1 .. 70

TABLE 25. Reduced form of Table 23 70

TABLE 26. Result of opposite changes in y_1 and y_2 70

TABLE 27. Result of sign analysis for Example 4.2.4 ... 83

TABLE 28. Sign table of $F=Q_F-Q_1-Q_2=0$ 91

TABLE 29. Sign table of $F=Q_1-C_{v1}\sqrt{P_F-P_1}=0$ 93

TABLE 30. Sign table of $F=Q_2-C_{v2}\sqrt{P_F-P_2}=0$ 94

TABLE 31. Pure sign table of $F=Q_1-C_{v1}\sqrt{P_F-P_1}=0$ 95

TABLE 32. Pure sign table of $F=Q_2-C_{v2}\sqrt{P_F-P_2}=0$ 96

TABLE 33. Result of Example 5.2.1 97

TABLE 34. Result of Example 5.2.2 98

TABLE 35. Result of Example 5.2.3 98

TABLE 36. Result of Example 5.2.6 100

TABLE 37. Result of Example 5.2.7 101

TABLE 38. Result of Example 5.2.8 101

TABLE 39. Reduced form of T_1 103

TABLE 40. Reduced form of T_2 103

TABLE 41. Reduced form of T_3 103

TABLE 42. Reduced form of Table 41 103

TABLE 43. Reduced sign table of gains for $F=Q_F-Q_1-Q_2=0$. 105

TABLE 44. Result of Example 5.3.1 107

TABLE 45. Result of Example 5.3.2 107

TABLE 46. Result of Example 5.3.3 107

I. INTRODUCTION

System analysis is an important topic for many scientific disciplines. By a system, we mean an orderly, interconnected arrangement of parts describing a scientific event. Almost everything in the world around us can be regarded as a system or as a part of a more complicated system. Therefore it is inevitable for science, whose main aim is to explain the real world truly, to base its grounds on the systematic analysis of systems.

The formal representation of systems is done through mathematical modeling. Thus the analysis of a system is performed by analysing the mathematical model of the system. We can classify this process into two: Quantitative analysis and qualitative analysis.

There has been tremendous work in the literature for the analysis of mathematical models. This work is almost entirely based on quantitative techniques and the literature is full of a rich set of theories for this purpose. For example, the dynamic systems theory (e.g. [1]) concerns with the quantitative solution of models that can be expressed in terms of difference and differential equations. Figure 1 shows the stages of the quantitative approach. Once the problem under consideration is defined and converted into a mathematical model, it is solved quantitatively by using several techniques. The solutions are then interpreted, the validity of the model is questioned, and the model is improved if necessary. The process of modeling, computation, and interpretation continues until the analyst is satisfied with the level of detail included in the model.

Besides these large amount of quantitative techniques, recently researchers have begun to focus their attention towards methods which are qualitative in nature. Artificial intelligence literature includes several approaches to study causal behavior of (especially physical) systems by using qualitative versions of quantitative methods, such as qualitative arithmetic, qualitative causal calculus, and qualitative physics. Figure 2 shows the stages involved in the qualitative approach. In this case we are saved from the

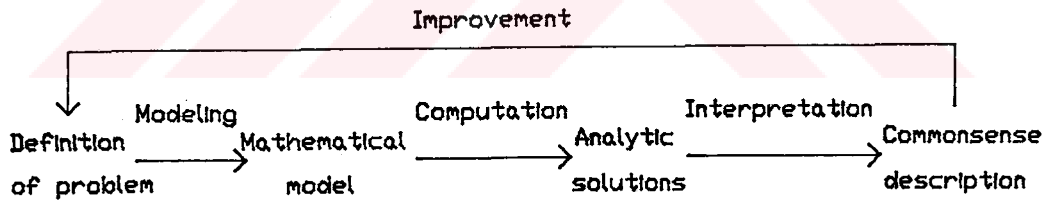

FIGURE 1. Stages of quantitative analysis

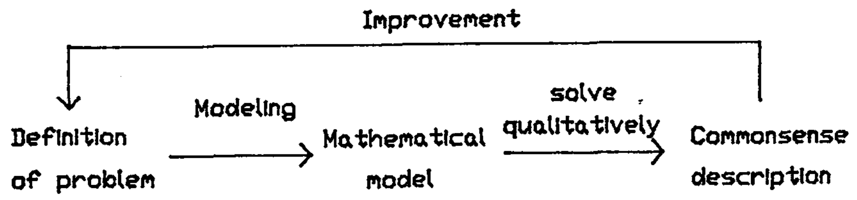

FIGURE 2. Stages of qualitative analysis

rather expensive computation step and the computation and the interpretation steps are replaced by a single step which solves the model qualitatively. What constitutes the phrase "solving qualitatively" is the topic of research.

The major advantage of a qualitative approach is that it parallels the way people solve problems in everyday life. It is well-known that a human being, when encountered with an analysis problem, does not try to generate the solutions quantitatively (which may indeed be very difficult or almost impossible for complex systems), instead he/she tries to capture the important parts of the problem, ignoring the irrelevant details, and makes deductions on these. This is referred to as *commonsense reasoning*. This explains why the field of artificial intelligence, whose ultimate goal is to simulate the functions of a human being, is interested in qualitative analysis.

1.1. Mathematical Models and Qualitative Analysis

Nearly all observed phenomena in our daily lives or in scientific investigation are based on mathematical models. Typical examples include physical, economical, and social systems as well as many others. It is absolutely essential that an adequate mathematical model of the system be generated if a meaningful understanding of the system is to be obtained.

A mathematical model is a specification of the interrelationships of the parts of a system, sufficiently explicit to enable us to study its behavior under a variety of circumstances and, in particular, to control it and to predict its future. There may be various models for the same system, differing in their amounts of detail. The degree of detail incorporated in the model depends on the choice of the model-builder, the amount of data available, and the application.

A major decision, often taken at the very onset of modeling, concerns the nature of the variables involved. Basically variables are of one of two types. Those variables whose values can be set freely by mechanisms that are outside the particular model under consideration are called exogenous variables. Those variables whose values are to be determined by the model are called endogenous variables. The exogenous variables are treated as independent variables and the endogenous variables are treated as dependent variables in the mathematical sense.

One important aspect of a mathematical model lies in the arrangement of the equations. It has been found by experience that if the equations are arranged in a logical or cause-and-effect sequence the model is stable. This sequence is termed the natural order, for it invariably closely parallels the cause-and-effect sequence found in nature. The key to understanding the internal mechanism lies in being able to define this natural cause-and-effect sequence.

Simultaneity, linearity, sufficiency and redundancy are some of the important properties of equations. The concept of simultaneity refers to the fact that the equation set must be solved simultaneously in order to determine the values of the dependent variables. Here we call an equation linear if the relationship between the variables can be shown as a straight line on the graph. In order to obtain a solution to a set of equations, it is necessary to specify as many independent equations as there are dependent variables. Independent equations mean that no redundant equation derived from the other equations can be used as an independent equation.

A mathematical model can be constructed using different kinds of mathematical notations, including algebraic equations, differential equations, difference equations, integral equations, and partial differential equations. For example, a model may be purely algebraic (that is, represented as a set of simultaneous algebraic equations), or a mixture of algebraic and differential equations.

There is a growing amount of research in the area of qualitative analysis of systems, although often under different titles (e.g. qualitative physics, qualitative process theory, qualitative simulation, commonsense knowledge, or causal ordering). All propose formalizing the commonsense knowledge about the everyday world, but each views this task rather differently.

The work by de Kleer and Brown [2,3] on qualitative physics is motivated by the desire to identify the core knowledge that underlies people's physical intuitions. Their interest in causality includes both the causal ordering among variables (what variables affect what other variables) and the direction, positive or negative, of causal effects (whether an increase in variable x causes the value of variable y to increase or decrease). They describe behavior by a set of qualitative differential equations called confluences. Confluences are constraint equations written in terms of qualitative derivatives of variables, whose values are +, -, or 0.

In the theory of qualitative physics as developed by de Kleer and Brown, two types of behavior are discussed: Intrastate behavior, or action within a state of the system; and interstate behavior, or the transformation of the system from one state to another. Boundaries between states are characterized by critical values of certain variables.

They propose a method, called mythical causality, to discover the causal relations in the system. The system, which is presumed to be at equilibrium initially, is disturbed by a change in the value of a variable and the effect of this disturbance is then propagated through the confluences until all the variables are assigned new values and equilibrium is restored. During propagation, if an equation is encountered in which there remains more than one variable whose values are unknown, then some assumptions are made to reduce the number of unknowns to one. These assumptions are based on some heuristic rules relevant to the properties of physical systems.

The key to mythical causality is the design of heuristics. They must be good enough to prevent the method from generating unreasonable solutions.

Iwasaki and Simon [4] propose a method for causality, somewhat different from that of de Kleer and Brown. It is called causal ordering. What causal ordering accomplishes is to make the asymmetric relationship among variables implied by the model explicit. Establishing a causal ordering involves finding subsets of variables whose values can be computed independently of the remaining variables, and using these values to reduce the structure to a smaller set of equations containing only the remaining variables. In this way, by knowing which variables can be computed outside the system and by using the modeling equations, a causal structure is generated for the system.

Kuipers [5,6] follows a quite different approach to describe the behavior of systems. His work is based on qualitative simulation which is a key inference process in qualitative causal reasoning. A system is described by a set of qualitative constraint equations and an initial state is given. Then qualitative simulation is used to predict the possible behaviors of the system. An important aspect of the method proposed by Kuipers is that it allows new qualitative values indicating critical points, in addition to +, -, and 0, to be discovered during the analysis. The basic idea behind qualitative simulation is that a parameter can change state only in specific directions defined by transition ordering rules and restricted by consistency filters. In this way a set of possible future behaviors of a system is generated.

Weld [7] describes comparative analysis, which can be seen as the complement of qualitative simulation, and proposes a method called differential qualitative (DQ) analysis. Whereas qualitative simulation takes a description of a system and predicts its behavior, comparative analysis takes as input this behavior and a perturbation and outputs a description of how and why the behavior would change as a result of the perturbation. DQ analysis uses inference rules to deduce qualitative information about the relative change of system parameters. A trace of the rules used in solving a problem is then translated into an intuitive explanation of the answer.

Qualitative process theory, developed by Forbus [8], is based on the concept of process. A process is defined as something that acts through time to change the parameters in a situation. The method reasons qualitatively about processes, when they will occur, their effects, and when they will stop. Studying naive physics reasoning about everyday physical situations, he determines the current set of active processes. The constraint equations are derived from the complete set of currently active processes. In this way, qualitative process theory provides the means to model aspects of commonsense reasoning about physical domains.

Weld [9] describes an abstraction technique, called aggregation, to dynamically create new descriptions of a system´s behavior. Aggregation works by detecting repeating cycles of processes and creating a continuous process description of the cycle´s behavior. The importance of the method is that it generates a higher-level abstraction for the behavior of a system by combining repeating processes.

In addition to these works, several other researchers have contributed to the literature for qualitative analysis, such as Hayes [10,11] and Williams [12]. As indicated at the beginning of this section, all these aim to formalize the commonsense knowledge, usually in different directions, such as deriving causal relations in a system, predicting how a system behaves, and explaining changes in system behavior.

1.2. Summary of Our Approach

The aim of this thesis is to analyse the mathematical model of a system and to generate causal relations about the dynamics of the system.

The mathematical model of a system contains implicit information about the system it describes - how the system behaves under various conditions, what are the relationships between the variables, how the cause-effect sequence of the variables is arranged, etc. Currently, the exposition of this information in order to understand the system truly is usually performed by humans. Our work is a step towards the automation of this process and introduces for this purpose some techniques. These techniques are based on the use of some well-known mathematical tools: partial derivatives and total differentials of closed form functions defining a system. Partial derivatives and total differentials are analysed to make the causal relations implied by the model explicit.

The mathematical models addressed by our work are restricted to models of the form of algebraic equations. Furthermore, it is assumed that the equation set constituting the model does not include any redundant equations. The equations may be linear or nonlinear. The extension of the techniques to other types of mathematical models (models expressed in terms of differential equations, difference equations, etc) is an area left for future research.

The outline of this thesis is as follows: Section 2 gives the definitions and the notation that will be used throughout the thesis and explains the fundamental assumption on which all this work is based. Sections 3 and 4 describe two techniques, referred to as sign analysis and value analysis. Section 5 illustrates what has been explained in previous sections by an example. Section 6 describes a prototype program which was implemented based on these techniques. Section 7 is a conclusion; it includes a comparison of our work with the previous works and discusses the possible future extensions of the techniques we have introduced. The appendices are reserved for the supplementary information about the program. Appendix A explains the Newton-Raphson method used in the program. Program messages are given in Appendix B. Appendix C is for the specifications and limitations of the program. In appendices D and E, a sample run and the listing of the program are given.

II. DEFINITIONS, NOTATION AND THE FUNDAMENTAL ASSUMPTION

This section explains the definitions and the notation that will be used throughout the thesis.

Definition 2.1. A *system*, in its broadest sense, is an orderly, interconnected arrangement of parts describing a scientific event.

This definition is general enough that nearly all natural phenomena can be described in the language of systems. Specific examples may arise, as well as many others, in (a) a physical system, such as a circuit, a traveling space vehicle, or a home heating system; (b) a social system, such as the behavior of an economic structure; or (c) a life system, such as population growth.

The techniques introduced in this work can well be applied to systems from any domain, although we mostly take our examples from physical systems.

Definition 2.2. A *subsystem* is a distinct conceptual part of a system. The working of the whole system is to be explained in terms of the functions of the subsystems that constitute the system.

As an example, a complex electrical circuit system can be divided into different subsystems. Each subsystem describes one distinct self-contained part of the system, and the principles about how the subsystems should communicate with each other are extracted.

This modular approach to mathematical modeling, that is combining subsystems to form a single system, is referred to as *assembling* of systems. It simplifies both the formulation and the analysis of the problem since it allows the analyst to investigate the system at different levels of detail. At the beginning the system can be viewed as a rough approximation, consisting of only a few subsystems, of the corresponding natural phenomenon by suppressing the details. Then as the analysis progresses, we can turn this system into a subsystem, assemble it with other relevant subsystems and thus form a more complicated system for a better approximation. The process continues in this way. At a particular stage of the process we can add more detail into a subsystem by breaking it into sub-subsystems or remove those subsystems that are proved to be inadequate in order to make the system more resemble to its real world counterpart.

Definition 2.3. A *mathematical model*[1] of a system (or of a subsystem) is the description of the system (or of the subsystem) in the mathematical language in terms of the following:

(1) A set of simultaneous equations, and

(2) A set of parameters.

[1]The term mathematical model will be abbreviated as model hereafter.

The terms *equation* and *function* will be used interchangeably throughout the thesis. A function denotes a relation between the parameters (see Definition 2.4) of the system and is written in the form of an equation while defining the model of the system.

As indicated in Section 1.2, the basic restriction imposed on the equations in this work is that the equations must be algebraic in form.

We will occasionally use the term *submodel* to indicate the mathematical model of a subsystem.

Definition 2.4. A *parameter* denotes either a variable or a constant. This is called the *type of a parameter*.

Definition 2.5. The *type of a variable* is either exogenous or endogenous. An *exogenous variable* is one whose value can be set freely by mechanisms that are outside the particular model under consideration. An *endogenous variable* is one whose value is to be determined by the interactions between the parameters in the model.

Exogenous variables and endogenous variables are sometimes called independent variables and dependent variables, respectively.

The choice of the type of a variable in a model depends on several factors, including the properties of the system under consideration, the level of detail incorporated into the model, and the taste of the model-builder. There is no clear-cut rules valid for all systems in this issue.

At this point it is worthwhile to state an important property of the variables which is applicable to most systems. A variable included in the model of a system which is termed as endogenous stays as endogenous if the model is enlarged by adding models for new subsystems. On the other hand, a variable which is termed as exogenous may either stay as exogenous or become endogenous after the same enlargement. This is basically due to the fact that, in the latter case, the value of the exogenous variable, which we can set freely, is now determined by a (newly incorporated) subsystem, thus the type of the variable becomes endogenous in the enlarged model. On the contrary, in the former case, the addition of a new subsystem does not enable us to set freely the value of an already endogenous variable.

A constant, as it name suggests, is a parameter of the model whose value remains constant throughout the analysis. In this work we treat constants as exogenous variables for all cases because of their similarity. Thus, in contrast to the above definition of a constant, we will talk about the "change" in the value of a constant. Also, as is the case for an exogenous variable, a constant is allowed to change its type to an endogenous variable in case new subsystems are added to the model. This approach is more flexible than fixing the value of a constant because a constant may be replaced by another one, similar to the change of the value of an exogenous variable, and it may be interesting to examine the variations in the system behavior in such a situation.

Definition 2.6. The difference between the number of parameters and the number of equations in a model is called the *degrees of freedom* of the model (or of the system).

Definition 2.7. A system whose degrees of freedom is equal to the number of exogenous variables and constants is called *singular* if, when the values of the exogenous variables and constants are given, the determinant of the resulting equation set is zero. The system is *nonsingular* otherwise.

Singularity is a mathematical (numerical) issue. It may arise for different reasons. A common reason is that a model may be wrong in the mathematical sense. That is, either the model contains redundant equations or a number of improper equations are added to the model in order to reduce the degrees of freedom of the system.

Definition 2.8. If the degrees of freedom of a system is equal to the number of exogenous variables and constants and the system is nonsingular, then the equations can be solved uniquely for the values of the endogenous variables for a given set of values for the exogenous variables and constants. The resulting set of values for the parameters is called the *state of the system*, or the *system state*. The system state is a complete description of the system.

Definition 2.9. The *operating region of a parameter* is the range of values that the parameter may take on.

Definition 2.10. The union of the operating regions of all the parameters in a model (or system) is called the *space of the system*, or the *system space*.

The degrees of freedom of a system being greater than the number of exogenous variables and constants denotes the number of additional equations that can be given about the system. This is usually done by fixing the values of a number of endogenous variables to reduce the degrees of freedom, thus making those variables exogenous. This must be done in a way such that the resulting system is nonsingular.

Definition 2.11. The *sign of* α, denoted as $[\alpha]$, where α may be a parameter, a total differential, a partial derivative, or an arithmetic expression, is defined as follows:

$[\alpha]=+$ iff $\alpha>0$,

$[\alpha]=0$ iff $\alpha=0$,

$[\alpha]=-$ iff $\alpha<0$.

Definition 2.12. The *inverse of the sign of* α, denoted as $[\bar{\alpha}]$, where α is as in the previous definition, is equal to the sign of the negative of α. That is, $[\bar{\alpha}]=[-\alpha]$.

Definition 2.13. The *direction of change of a parameter* p, denoted as {p}, indicates in which direction the value of p changes. It is defined as follows:

$\{p\}=+$ iff the value of p increases,

$\{p\}=0$ iff the value of p does not change,

$\{p\}=-$ iff the value of p decreases.

Since the total differential of a parameter p, represented as dp, indicates mathematically the magnitude of the change in the value of p, we can state that $\{p\}=[dp]$. In other words, p is increasing iff $dp>0$, the value of p does not change iff $dp=0$, and p is decreasing iff $dp<0$.

Definition 2.14. The *inverse of the direction of change of a parameter* p, denoted as $\{\overline{p}\}$, is defined as follows:

$\{\overline{p}\}=+$ iff $\{p\}=-$,

$\{\overline{p}\}=0$ iff $\{p\}=0$,

$\{\overline{p}\}=-$ iff $\{p\}=+$.

Definition 2.15. A function is said to be in *closed form* if it is written in such a way that the value of the function is equal to zero.

An ordinary function can easily be converted to a closed form function. For example, the closed form of the function $y=x$ is $y-x=0$.

Below is a listing of the notation that will be used throughout the thesis:

x_i an endogenous variable

y_i an exogenous variable or a constant

z_i a parameter (i.e. either an endogenous variable, an exogenous variable, or a constant).

dz_i total differential of the parameter z_i.

$F_i(z_1,\ldots,z_n)$ a function of parameters $z_1,\ldots,z_n$. The parameter list will be dropped and the function symbol will be denoted simply as F_i when no confusion arises.

$dF_i(z_1,\ldots,z_n)$ total differential of the function F_i (abbreviated as dF_i).

$\frac{\partial F_i(z_1,\ldots,z_n)}{\partial p_j}$ partial derivative of the function F_i with respect to (wrt) the parameter p_j (abbreviated as $\frac{\partial F_i}{\partial p_j}$).

$\frac{\partial z_i}{\partial z_j}$ partial derivative of parameter z_i wrt the parameter z_j.

$[\alpha]$ the sign of α (Definition 2.11).

$\{z\}$ the direction of change of the parameter z (Definition 2.13). The braces {} will also be used to represent the elements in a set. The distinction will be clear from the context.

$\bar{s}$ the inverse of s, where s represents either a sign or a direction of change. The distinction will be clear from the context.

At this point, the definitions given above will be illustrated by an example.

Consider the system shown in Figure 3, which illustrates a simple valve. The model of the system can be described as follows[2]:

[2] We will describe a model informally by listing the equations and parameters. For each parameter, its type (in abbreviated form: exo for exogenous variable, endo for endogenous variable, and const for constant) and (if applicable) a brief explanation are displayed.

Equations

$Q=C_v\sqrt{P_1-P_2}$

Parameters

Q ,	flow rate (endo)
C_v ,	valve constant (const)
P_1 ,	pressure at the left-hand side of the valve (exo)
P_2 ,	pressure at the right-hand side of the valve (exo)

It is assumed that P_1 is greater than P_2 and thus the flow is from left to right. (P_1-P_2) is called the pressure drop across the valve.

There is one equation and four parameters. The degrees of freedom of the system is three, thus the equation can be solved to determine uniquely the value of Q given the values of other parameters.

Since C_v represents the valve constant, physically it can take on only positive values. Therefore we can denote the operating region of C_v by the interval (0,c] for some value c.

Now consider the system shown in Figure 4, which is an extension of the previous system in having a vessel

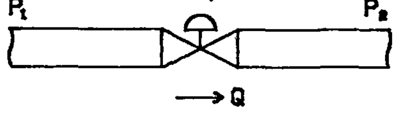

FIGURE 3. A valve system

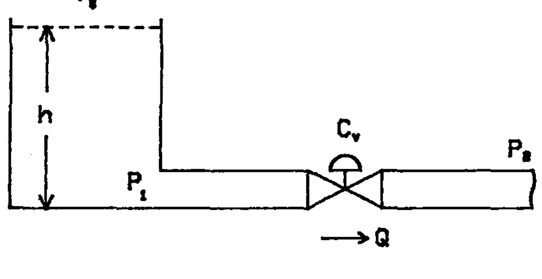

FIGURE 4. The valve attached to a vessel

containing liquid at the left-hand side. The model of this new system can be described as follows:

Equations

$Q = C_v \sqrt{P_1 - P_2}$

$P_1 = h\theta + P_0$

Parameters

Q ,	flow rate (endo)
C_v,	valve constant (const)
P_1,	pressure at the left-hand side of the valve and at the bottom of the liquid (endo)
P_2,	pressure at the right-hand side of the valve (exo)
h ,	height of liquid (exo)
θ ,	density of liquid (const)
P_0,	pressure above liquid surface (exo)

Note that the type of the variable P_1 is changed from exogenous to endogenous. This means that in the first system the value of P_1 is set freely by the analyst in order to find the value of Q, but in the extended system P_1 is determined by the model itself (i.e., by the second equation, given the values of h, θ, and P_0) which in turn determines Q given C_v and P_2.

The system can be thought of as being composed of two subsystems, say the valve system and the liquid system. We can construct the models of these two subsystems separately and then assemble them to form a single system by defining the mappings between the parameters. The model of the valve system is given above; the liquid system can be modeled as follows:

Equations

$P=h\theta+P_o$

Parameters

P , pressure at the bottom (endo)

h , height of liquid (exo)

θ , density of liquid (const)

P_o, pressure above liquid surface (exo)

The two can then be merged into a single system by indicating that the type of P_1 changes to endogenous and P_1 of the valve model and P of the liquid model denote the same quantity.

The techniques introduced here are based on a fundamental assumption: A relation expressed in the form of an algebraic equation between the parameters of a system implicitly represents the causal relations between the parameters as well.

Consider the following closed form function

$$F(x,y)=x+y=0 \qquad (2.1)$$

Suppose that x is an endogenous variable and y is an exogenous variable. It is customary to write the equations in a model explicitly for endogenous variables to denote the fact that the endogenous variables are to be computed from the model for a given set of values for the exogenous variables. Following this recipe, the relationship between the two variables in the above function can be shown more clearly in the following form

$$x=F(y)=-y$$

which states that the endogenous variable x is a function of the exogenous variable y and the function maps the negative of y to x. This relation between x and y may also be interpreted that if the value of y is increased then the value of x will decrease to satisfy the equation and vice versa. This interpretation of the function implies the following causal relations between x and y:

an increase in the value of y causes a decrease in the value of x.
a decrease in the value of y causes an increase in the value of x.

These relations may be rewritten as follows using an *if/then* construct keeping in mind that the *if/then* construct represents causality.

if y changes in positive direction then x changes in negative direction
if y changes in negative direction then x changes in positive direction

On the other hand, considering the form of the function shown in (2.1) and assuming that we have imposed no distinction between the variables from the point of being either endogenous or exogenous, it is possible to treat the variables in a uniform way and to generate the following four relations between the variables:

if y changes in positive direction then x changes in negative direction
if y changes in negative direction then x changes in positive direction
if x changes in positive direction then y changes in negative direction
if x changes in negative direction then y changes in positive direction

Note that two of the relations above would be invalid if one of the variables was indeed an endogenous variable as in the first case. This means that we will treat all of the variables as if they are all exogenous variables for the sake of uniformity, but then we will make use of any information available on their being endogenous or exogenous. In case there is no such information we will assume that any of the variables can be changed freely. In this case all four of the relations will be assumed to hold and the relations will be interpreted as representing mathematical causalities.

III. SIGN ANALYSIS

The first technique that will be introduced here is called sign analysis. Sign analysis examines how a system will react to perturbations in its parameters.

The idea can be explained as follows: The model of a system is given. This model may only contain the equations and the parameter names. It is not necessary to state whether the type of a parameter is variable or constant and whether the type of a variable is exogenous or endogenous, since the sign analysis technique treats all the parameters in a uniform way and leaves the interpretation of the results to the analyst. The values of the parameters (i.e. a particular system state) need not be known, either.

When the model to be examined is given, we state a perturbation in one or more of the parameters. In other words, we state the directions of change of those parameters. The model is then analysed and the result is the directions of change of the remaining parameters.

We will examine the application of the technique on two quantities: Sign analysis of total differentials and sign analysis of gains. The technique is based on the use of what we name as sign tables, which is the topic of the following section.

3.1. Sign Tables

In this section we will examine in detail the sign tables for total differentials of closed form functions. In Section 3.3, we will briefly discuss the application of sign tables to gains.

For each equation (function) in a model, we form a two-dimensional matrix called the *sign table of the equation (function)*. The sign table of an equation has as many columns as the number of parameters in the equation. A column for a parameter z is titled with the sign of the total differential of z, that is [dz], and each element under the column is the sign of a partial derivative representing a sign for the total differential of z. Since we know that [dz]={z}, the elements can also be interpreted as denoting the directions of change of the parameter z. Each row represents a combination of the signs of total differentials of the parameters (i.e. directions of change of the parameters) in the equation. The number of rows in a sign table depends on the number of parameters in the equation, as will be described below.

Let us consider the following closed form function of two parameters

$$F(z_1,z_2)=0$$

The total differential of this function expressed symbolically is

$$dF=\frac{\partial F}{\partial z_1}\,dz_1+\frac{\partial F}{\partial z_2}\,dz_2=0$$

Note that dF is equal to zero since it is the total differential of the constant zero, i.e. the change in the value of the function (dF) must be zero regardless of the changes in its parameters. For simplicity, let us denote the signs of the partial derivatives $\frac{\partial F}{\partial z_1}$ and $\frac{\partial F}{\partial z_2}$, $[\frac{\partial F}{\partial z_1}]$ and $[\frac{\partial F}{\partial z_2}]$, as s_1 and s_2, respectively. Then possible combinations for the signs $[dz_1]$ and $[dz_2]$ of the parameters z_1 and z_2 may be given in the following sign table:

$[dz_1]$	$[dz_2]$
s_1	$\bar{s}_2$
$\bar{s}_1$	s_2
0	0

As indicated above, [dz] denotes the same identity as {z}. Therefore we could as well title the columns in the above table as $\{z_1\}$ and $\{z_2\}$ instead of $[dz_1]$ and $[dz_2]$.

Note that the elements in the table correspond to the signs of partial derivatives.

Each row of the table represents a possible change in the values of the parameters. The first row says that a change in z_1 in the same direction as s_1 and a change in z_2 in the opposite direction of s_2 is compatible with the model. Likewise, the second row corresponds to a change in z_1 in the opposite direction of s_1 and a change in z_2 in the same direction as s_2; and the third row corresponds to a change in none of the parameters.

If we consider that each parameter has three possible directions of change, there are 3^n sign combinations for a function of n parameters. This corresponds to nine for a two-parameter function but as indicated above and as will be derived below only three of these are valid. For example, a change in z_1 in the same direction as s_1 and a change in z_2 in the same direction as s_2 is not possible.

It is important to emphasize that the above discussion is valid regardless of the values of the parameters, i.e. regardless of the system state.

Now let us give an example to make the point clearer. Consider the following two-parameter closed form function

$$F(z_1,z_2)=z_1+z_2=0$$

The total differential of the function is

$$dF=\frac{\partial F}{\partial z_1}\,dz_1+\frac{\partial F}{\partial z_2}\,dz_2=0$$

The partial derivatives are

$$\frac{\partial F}{\partial z_1}=1 \qquad \frac{\partial F}{\partial z_2}=1$$

Putting these into the above equality, it becomes

$$dF=dz_1+dz_2=0$$

Since $s_1=[\frac{\partial F}{\partial z_1}]=+$ and $s_2=[\frac{\partial F}{\partial z_2}]=+$, the table for a two-parameter function given above in its general form reduces to the following particular table for this example

$[dz_1]$	$[dz_2]$
+	−
−	+
0	0

Thus we conclude that the parameters of the function can change in one of three ways (regardless of the values of z_1 and z_2): z_1 increases and z_2 decreases, z_1 decreases and z_2 increases, or z_1 and z_2 remain unchanged. The remaining sign combinations are not valid. For example, it is not possible for both z_1 and z_2 to increase (that is, a change in z_1 in the same direction as s_1=+, and a change in z_2 in the same direction as s_2=+) and still the equation to hold. This can also be easily seen from the function.

Tables 1, 2, 3, and 4 give the sign tables for functions of one, two, three, and four parameters, respectively. As in the above discussion, s_i denotes $[\frac{\partial F}{\partial z_i}]$ for the parameter z_i.

TABLE 1. Sign table of $F(z)=0$

$[dz]$
0

TABLE 2. Sign table of $F(z_1,z_2)=0$

$[dz_1]$	$[dz_2]$
s_1	$\bar{s}_2$
$\bar{s}_1$	s_2
0	0

TABLE 3. Sign table of $F(z_1,z_2,z_3)=0$

$[dz_1]$	$[dz_2]$	$[dz_3]$
s_1	s_2	$\bar{s}_3$
s_1	$\bar{s}_2$	s_3
s_1	$\bar{s}_2$	$\bar{s}_3$
s_1	$\bar{s}_2$	0
s_1	0	$\bar{s}_3$
$\bar{s}_1$	s_2	s_3
$\bar{s}_1$	s_2	$\bar{s}_3$
$\bar{s}_1$	s_2	0
$\bar{s}_1$	$\bar{s}_2$	s_3
$\bar{s}_1$	0	s_3
0	s_2	$\bar{s}_3$
0	$\bar{s}_2$	s_3
0	0	0

TABLE 4. Sign table of $F(z_1,z_2,z_3,z_4)=0$

$[dz_1]$	$[dz_2]$	$[dz_3]$	$[dz_4]$	$[dz_1]$	$[dz_2]$	$[dz_3]$	$[dz_4]$
s_1	s_2	s_3	$\bar{s}_4$	$\bar{s}_1$	s_2	0	$\bar{s}_4$
s_1	s_2	$\bar{s}_3$	s_4	$\bar{s}_1$	s_2	0	0
s_1	s_2	$\bar{s}_3$	$\bar{s}_4$	$\bar{s}_1$	$\bar{s}_2$	s_3	s_4
s_1	s_2	$\bar{s}_3$	0	$\bar{s}_1$	$\bar{s}_2$	s_3	$\bar{s}_4$
s_1	s_2	0	$\bar{s}_4$	$\bar{s}_1$	$\bar{s}_2$	s_3	0
s_1	$\bar{s}_2$	s_3	s_4	$\bar{s}_1$	$\bar{s}_2$	$\bar{s}_3$	s_4
s_1	$\bar{s}_2$	s_3	$\bar{s}_4$	$\bar{s}_1$	$\bar{s}_2$	0	s_4
s_1	$\bar{s}_2$	s_3	0	$\bar{s}_1$	0	s_3	s_4
s_1	$\bar{s}_2$	$\bar{s}_3$	s_4	$\bar{s}_1$	0	s_3	$\bar{s}_4$
s_1	$\bar{s}_2$	$\bar{s}_3$	$\bar{s}_4$	$\bar{s}_1$	0	s_3	0
s_1	$\bar{s}_2$	$\bar{s}_3$	0	$\bar{s}_1$	0	$\bar{s}_3$	s_4
s_1	$\bar{s}_2$	0	s_4	$\bar{s}_1$	0	0	s_4
s_1	$\bar{s}_2$	0	$\bar{s}_4$	0	s_2	s_3	$\bar{s}_4$
s_1	$\bar{s}_2$	0	0	0	s_2	$\bar{s}_3$	s_4
s_1	0	s_3	$\bar{s}_4$	0	s_2	$\bar{s}_3$	$\bar{s}_4$
s_1	0	$\bar{s}_3$	s_4	0	s_2	$\bar{s}_3$	0
s_1	0	$\bar{s}_3$	$\bar{s}_4$	0	s_2	0	$\bar{s}_4$
s_1	0	$\bar{s}_3$	0	0	$\bar{s}_2$	s_3	s_4
s_1	0	0	$\bar{s}_4$	0	$\bar{s}_2$	s_3	$\bar{s}_4$
$\bar{s}_1$	s_2	s_3	s_4	0	$\bar{s}_2$	s_3	0
$\bar{s}_1$	s_2	s_3	$\bar{s}_4$	0	$\bar{s}_2$	$\bar{s}_3$	s_4
$\bar{s}_1$	s_2	s_3	0	0	$\bar{s}_2$	0	s_4
$\bar{s}_1$	s_2	$\bar{s}_3$	s_4	0	0	s_3	$\bar{s}_4$
$\bar{s}_1$	s_2	$\bar{s}_3$	$\bar{s}_4$	0	0	$\bar{s}_3$	s_4
$\bar{s}_1$	s_2	$\bar{s}_3$	0	0	0	0	0
$\bar{s}_1$	s_2	0	s_4				

In the following proposition we will explain the mathematical foundation of sign tables.

Proposition 3.1. The sign table of an equation includes *all* the possible directions of change of the parameters in the equation.

Proof We give the proof for the general case. Consider the function F of n parameters $z_1,...,z_n$

$$F(z_1,...,z_n)=0$$

The total differential of the function is

$$dF=\frac{\partial F}{\partial z_1}\,dz_1+...+\frac{\partial F}{\partial z_n}\,dz_n=0 \qquad (3.1)$$

By referring to the mathematical meaning of partial derivatives and total differentials we can state the following facts: $[\frac{\partial F}{\partial z_i}]=+$ (i.e. $\frac{\partial F}{\partial z_i}>0$) means that the value of the function F changes in the same direction as the value of the parameter z_i; that is, either increasing z_i causes F to increase or decreasing z_i causes F to decrease. $[\frac{\partial F}{\partial z_i}]=-$ ($\frac{\partial F}{\partial z_i}<0$) means that F changes in the opposite direction of z_i; that is, either increasing z_i causes F to decrease or decreasing z_i causes F to increase. We do not consider the case $[\frac{\partial F}{\partial z_i}]=0$ ($\frac{\partial F}{\partial z_i}=0$), since z_i is a parameter that appears in F with a nonzero coefficient, thus the partial derivative of F wrt z_i cannot be zero.

Now consider the total differentials of parameters. $[dz_i]=+$ ($dz_i>0$) means that the value of the parameter z_i is increasing. $[dz_i]=-$ $(dz_i<0)$ means that z_i is decreasing. $[dz_i]=0$ $(dz_i=0)$ means that z_i does not change.

If $[\frac{\partial F}{\partial z_i}]=[dz_i]$, then $[\frac{\partial F}{\partial z_i}\, dz_i]=+$. In other words, if both $\frac{\partial F}{\partial z_i}$ and dz_i are positive or both of them are negative, then the multiplication of these two quantities is positive. This means that a change in z_i, no matter whether it is an increase or a decrease, causes F to increase. If $[\frac{\partial F}{\partial z_i}]=[dz_i]=+$, then z_i is increasing and F and z_i change in the same direction, thus F increases. If $[\frac{\partial F}{\partial z_i}]=[dz_i]=-$, then z_i is decreasing and F and z_i change in the opposite directions, thus F increases. If $[\frac{\partial F}{\partial z_i}]=[\overline{dz_i}]$, then $[\frac{\partial F}{\partial z_i}\, dz_i]=-$. In other words, if one is positive and the other is negative,then the multiplication is negative. This means that a change in z_i, no matter whether it is an increase or a decrease, causes F to decrease. If $[\frac{\partial F}{\partial z_i}]=+$ and $[dz_i]=-$, then z_i is decreasing and F and z_i change in the same direction, thus F decreases. If $[\frac{\partial F}{\partial z_i}]=-$ and $[dz_i]=+$, then z_i is increasing and F and z_i change in the opposite directions, thus F decreases. If $[dz_i]=0$, then $[\frac{\partial F}{\partial z_i}\, dz_i]=0$. In other words, if the total differential of z_i is zero, then the multiplication is zero regardless of the value of $\frac{\partial F}{\partial z_i}$.

We require the total differential shown in (3.1) be equal to zero. In order for this equality to hold, either none of the parameters change or there are at least two parameters, one causing F to increase and one causing F to decrease. In the former case, $[dz_i]=0$ $(dz_i=0)$, so $[\frac{\partial F}{\partial z_i} dz_i]=0$ $(\frac{\partial F}{\partial z_i} dz_i=0)$ for each i=1,...,n and thus the equality holds. The row of the sign table corresponding to this possibility contains $[dz_i]=0$ for each i=1,...,n. In the latter case, there are at least two parameters, say z_i and z_j, $i\neq j$, such that $[\frac{\partial F}{\partial z_i} dz_i]=+$ $([\frac{\partial F}{\partial z_i}]=[dz_i])$ and $[\frac{\partial F}{\partial z_j} dz_j]=-$ $([\frac{\partial F}{\partial z_j}]=[\overline{dz_j}])$. The row of the table corresponding to this possibility contains $[dz_i]=[\frac{\partial F}{\partial z_i}]$ and $[dz_j]=[\overline{\frac{\partial F}{\partial z_j}}]$. There is exactly one row in each table corresponding to the former case, usually shown as the last row of the table. The other rows correspond to the second case. □

As can be seen from Tables 1 through 4, all the rows are in accordance with the definition given in the proof of the proposition. For example, consider the sign table for a two-parameter function given in Table 2 and the table, given in Table 5, of the same function showing all of the nine sign combinations. Each of the six rows that appear in Table 5 but not in Table 2 has the common property that either it contains at least one element in the same direction of the corresponding partial derivative but lacks of an element in the opposite direction, or vice versa. For instance, the first row of Table 5 says that both of the variables change in the same directions as the corresponding partial derivatives. That is, $[\frac{\partial F}{\partial z_1}]=[dz_1]$ and $[\frac{\partial F}{\partial z_2}]=[dz_2]$. So,

TABLE 5. Table of all sign combinations for $F(z_1,z_2)=0$

$[dz_1]$	$[dz_2]$
s_1	s_2
s_1	$\bar{s}_2$
s_1	0
$\bar{s}_1$	s_2
$\bar{s}_1$	$\bar{s}_2$
$\bar{s}_1$	0
0	s_2
0	$\bar{s}_2$
0	0

$[\frac{\partial F}{\partial z_1} dz_1]=+$ and $[\frac{\partial F}{\partial z_2} dz_2]=+$. This means that $\frac{\partial F}{\partial z_1} dz_1>0$ and $\frac{\partial F}{\partial z_2} dz_2>0$. Putting these quantities into the total differential of F (where the notation (+) indicates that the quantity is positive),

$$dF=\frac{\partial F}{\partial z_1} dz_1+\frac{\partial F}{\partial z_2} dz_2=0$$
$$= \quad (+) + (+) \quad =0$$

leads to an inconsistency since the addition of two positive quantities cannot be equal to zero.

Figure 5 gives a Pascal procedure for constructing the sign table of a function. The inputs to the procedure are the number of parameters in the function and the partial derivatives of the function wrt the parameters. We left how to represent the derivatives unspecified. The array *Mode* indicates whether the sign of the partial derivative, the

```
Const
  MaxPar = ... ;     { maximum number of parameters }

Type
  ParRange = 1 .. MaxPar ;     { parameter range }
  Derivative = Array [ParRange] Of ... ;
      { partial derivatives of the function wrt parameters }
      { each element of the array is an expression         }

Procedure SignTable (n     : ParRange  ;       { number of parameters in the function }
                     Deriv : Derivative ) ;    { partial derivatives for the function }
Var
  Mode : Array [ParRange] Of 1..3 ;
      { modes of parameters in the following sense : }
      { for a parameter z,                           }
      {   1 : [dz] = sign of partial derivative of the function wrt z }
      {   2 : [dz] = 0                                                }
      {   3 : [dz] = inverse of the sign of partial derivative of the function wrt z }
  i : ParRange ;

  Function ModesConsistent : Boolean ;
    { this function checks whether the modes of the parameters are valid, i.e. }
    { either all must be 2 or there must be at least two parameters, one       }
    { having mode 1 and the other having mode 3                                }
  Var
    ModeSet : Set Of 1..3 ;
    i : ParRange ;
  Begin
    ModeSet :=[] ;
    For i:=1 To n Do
      ModeSet := ModeSet + [Mode [i]] ;    { ModeSet holds the union of modes }
    ModesConsistent := (ModeSet * [1,3] = []) Or (ModeSet * [1,3] = [1,3])
  End ;

  Procedure DisplayRow ;
    { this procedure outputs a row of the table }
  Var
    i : ParRange ;
  Begin
    For i:=1 To n Do
      Case Mode [i] Of
        1 : Write (Deriv [i]) ;
        2 : Write (0) ;
        3 : Write (Inverse of Deriv [i])
      End
  End ;

Begin
  Mode [1] := 0 ;    { initialize mode of first parameter }
  i:= 1 ;            { begin with the first parameter     }
  While (i > 0) Do
    If Mode [i] < 3 Then        { continue with the next mode of the parameter }
    Begin
      Mode [i] := Mode [i] + 1 ;
      While (i < n) Do          { initialize modes of following parameters }
      Begin
        i := i + 1 ;
        Mode [i] := 1
      End ;
      If ModesConsistent Then
        DisplayRow
    End
    Else                         { all modes of the parameter examined, }
      i := i - 1                 { continue with the previous parameter }
End ;
```

FIGURE 5. A procedure for constructing the sign table of a function

inverse of the sign of the partial derivative, or the sign 0 will be used as an element of the table. We begin with mode 1 for a parameter and increase its mode until reaching 3. The procedure generates all 3^n sign combinations for a function with n parameters, which are then filtered by the function *ModesConsistent* to agree with the condition defined in the proof of Proposition 3.1. The procedure *DisplayRow* outputs a row of the table in an informal way.

The number of rows in the table of a function depends only on the number of parameters in the function. Therefore two functions with the same number of parameters will have tables with the same number of rows. The formula to calculate the number of rows in the table of a function is

$$N = \sum_{i=2}^{n} \binom{n}{i} (2^i - 2) + 1$$

where

N : number of rows in the table

n : number of parameters in the function.

The calculation is shown in Figure 6 for functions up to 10 parameters.

n	1	2	3	4	5	6	7	8	9	10
N	1	3	13	51	181	603	1933	6051	18661	57003

FIGURE 6. Number of rows, N, in the sign table of an n-parameter function

One important point that must be indicated here is that, since the elements in a sign table correspond to the signs of partial derivatives, we must know in advance these signs before starting the analysis in order to be able to reduce the sign table to one in which each element is either +, 0, or -. The expression of a partial derivative may or may not directly lead to a sign symbol when we take the partial derivative. In such a case, it is necessary to supply sufficient information about the parameters appearing in the partial derivative in order to find the sign of the partial derivative. This does not mean that we must specify the state of the system (i.e. the values of the parameters), instead we restrict the system space.

As an example, suppose that the partial derivative of a function F with respect to a parameter z is found to be

$$\frac{\partial F}{\partial z}=z-2$$

In order to be able to specify $[\frac{\partial F}{\partial z}]$ uniquely, we must know whether $z>2$, $z=2$, or $z<2$. These three possibilities correspond to $[\frac{\partial F}{\partial z}]$ being +, 0, or -, respectively. By making such a statement, we restrict the system space. For instance, if we say that $z>2$, this amounts to leaving nearly half of the operating region of z out of consideration (assuming that z can take on all the values on the real line) and the result of the sign analysis will only be valid for this region of the space.

On the other hand, if a partial derivative evaluates to a constant such as

$$\frac{\partial F}{\partial z}=1$$

or to an expression whose sign can be calculated directly such as

$$\frac{\partial F}{\partial z}=z^2+1$$

then we do not require any information about that parameter.

In most systems, a large number of partial derivatives would either be constants or their signs would be readily determined from their expressions for most functions even though the values of the parameters are not specified. This would greatly simplify sign analysis. For example, consider the following equation

$$F=P_1-h\theta-P_o=0$$

which was used in Section 2 for the valve and liquid model. The partial derivatives are

$$\frac{\partial F}{\partial P_1}=1 \qquad \frac{\partial F}{\partial h}=-\theta \qquad \frac{\partial F}{\partial \theta}=-h \qquad \frac{\partial F}{\partial P_o}=-1$$

Since only positive values are meaningful for the parameters θ (density of the liquid) and h (height of the liquid) in the system they take place (i.e. operating regions of θ and h do not contain negative values), the signs of all four of the partial derivatives would be easily determined without imposing any restrictions on the system space.

It must be noted that even though we are sometimes forced to partition the system space into regions in order to be able to evaluate the signs of partial derivatives, we can perform sign analysis on different regions of the system space and then merge the individual results into results which are valid for the whole system space (or valid for a subset of the system space which we are satisfied with).

As a final point, if one is interested in performing sign analysis for a particular state of the system, then the values of the partial derivatives will be evaluated first, from which their signs will be determined.

The above discussion leads us to the following definition.

Definition 3.1. A *pure sign table* is a sign table in which each element corresponds to a sign symbol, i.e. +, 0, or –. We can convert the sign table of a function into a pure sign table by supplying information about the signs of parameters and expressions in the table and computing the signs of those expressions.

3.2. Sign Analysis of Total Differentials

Sign analysis is based on the fact that the equations in the model satisfy a particular system state at a time. We can disturb this state in specific directions and the equations reach a new state. In between the old and the new states, the closed form equations always evaluate to zero regardless of in which direction the system is disturbed. The idea behind sign analysis is that it always keeps this principle in mind. When a parameter is being changed, it changes another parameter in such a way that the equations always hold.

Consider a set of m functions $F_1,\dots,F_m$ of n parameters $z_1,\dots,z_n$ as given below

$$\begin{aligned} &F_1(z_1,\dots,z_n)=0 \\ &\vdots \\ &F_m(z_1,\dots,z_n)=0 \end{aligned} \tag{3.2}$$

where $n \geq m$. To perform sign analysis on such a system, we first form the sign table of each function showing the possible sign combinations. (Although we have written the equations as though each function is a function of n parameters, a function need not contain all the parameters. The table of an equation will only contain columns for the parameters it includes). Given a perturbation in the system (i.e. directions of change for some of the parameters), we join the individual tables (see Definition 3.3) with respect to the perturbation to obtain the result of the analysis.

Definition 3.2. Suppose we have a model consisting of a set of m functions $F_1,\dots,F_m$ and n parameters $z_1,\dots,z_n$ as given in (3.2). A *perturbation formula*, denoted by θ, for the model is a logical formula in the form of $\{r_1\}=s_1 \wedge \{r_2\}=s_2 \wedge \dots \wedge \{r_k\}=s_k$, where $0 \leq k \leq n$; $r_i \in \{z_1,\dots,z_n\}$,

$i=1,\dots,k$; $r_i=r_j$ iff $i=j$, $i,j=1,\dots,k$; and s_i represents a direction of change symbol, i.e. $s_i \in \{+,0,-\}$, $i=1,\dots,k$. Informally, a perturbation formula is the logical conjunction of the directions of change for some of the parameters.

Definition 3.3. Suppose that we have two functions

$$F_1(p_1,\dots,p_n)=0$$

and

$$F_2(q_1,\dots,q_m)=0$$

where $p_1,\dots,p_n$ are parameters occurring in F_1 and $q_1,\dots,q_m$ are parameters occurring in F_2. p_i and q_j may denote the same parameter for any $i=1,\dots,n$ and $j=1,\dots,m$. Suppose that we represent the pure sign tables of F_1 and F_2 by the symbols T_1 and T_2, respectively. Let θ be a perturbation formula for F_1 and F_2, i.e. a conjunction of the directions of change for some of $p_1,\dots,p_n,q_1,\dots,q_m$. Then the *$\theta$-join of sign tables T_1 and T_2*, denoted as $T_1 \bowtie_{\theta} T_2$, is defined as follows:

(1) Form the cartesian product of T_1 and T_2, i.e. combine each row of T_1 with all the rows of T_2. Call the resulting table T_1'.

(2) Remove those rows of T_1' which contain different directions of change for the same parameter. That is, $[dr_i]=s_i$ and $[dr_j]=s_j$ are two different columns on the same row such that $r_i=r_j$ and $s_i \neq s_j$. Call the result T_2'.

(3) For each column of T_2', if there is another column for the same parameter then project out (delete) one of those columns. Call the result T_3'.

(4) Remove those rows of T_3' which do not agree with θ. A row does not agree with θ if θ contains a conjunct $\{r_i\}=s_i$ and the row contains a column $[dr_j]=s_j$ such that $r_i=r_j$ and $s_i \neq s_j$. The result is the θ-join of T_1 and T_2.

Property 3.1. For any sign tables T_1, T_2, and T_3, and any θ,

(1) $T_1 \bowtie_\theta T_2 = T_2 \bowtie_\theta T_1$ (commutativity of join)

(2) $(T_1 \bowtie_\theta T_2) \bowtie_\theta T_3 = T_1 \bowtie_\theta (T_2 \bowtie_\theta T_3)$ (associativity of join).

As an example, Figure 7 shows two sign tables. Figure 8 is the θ-join of these tables where $\theta=(\{z_1\}=+)$.

Let us call the join operation $T_1 \bowtie_\theta T_2$ where θ is empty, the *$\in$-join of T_1 and T_2*, and represent it by the notation $T_1 \bowtie_\in T_2$. Note that the first three steps of the above definition do not make use of θ. θ is used only in the fourth step. If θ is empty then the fourth step will have no function. Therefore $T_1 \bowtie_\in T_2$ also represents an operation which is defined by the first three steps of the above definition.

$[dz_1]$	$[dz_2]$
+	+
–	–
0	0

$[dz_2]$	$[dz_3]$
+	–
–	+
0	0

FIGURE 7. Two sign tables for the join example

$[dz_1]$	$[dz_2]$	$[dz_2]$	$[dz_3]$
+	+	+	−
+	+	−	+
+	+	0	0
−	−	+	−
−	−	−	+
−	−	0	0
0	0	+	−
0	0	−	+
0	0	0	0

Step 1

$[dz_1]$	$[dz_2]$	$[dz_2]$	$[dz_3]$
+	+	+	−
−	−	−	+
0	0	0	0

Step 2

$[dz_1]$	$[dz_2]$	$[dz_3]$
+	+	−
−	−	+
0	0	0

Step 3

$[dz_1]$	$[dz_2]$	$[dz_3]$
+	+	−

Step 4

FIGURE 8. Example of join operation

Definition 3.4. Consider a model whose functions and parameters are given as in (3.2). The *Information matrix*, denoted by I, of this model is defined as follows:

$$I = \begin{array}{c} \\ F_1 \\ \vdots \\ F_m \end{array} \begin{array}{c} \begin{array}{ccc} z_1 & \cdots & z_n \end{array} \\ \left[\begin{array}{ccc} \dfrac{\partial F_1}{\partial z_1} & \cdots & \dfrac{\partial F_1}{\partial z_n} \\ \vdots & & \vdots \\ \dfrac{\partial F_m}{\partial z_1} & \cdots & \dfrac{\partial F_m}{\partial z_n} \end{array} \right] \end{array}$$

That is, the Information matrix is the matrix of partial derivatives of functions wrt parameters. When the above matrix is formed from partial derivatives of functions wrt endogenous variables only, we will refer to it as the *Jacobian matrix* of the system and denote by J.

Definition 3.5. A *sign matrix* is a matrix in which each element represents a sign.

We will denote by I_s the sign matrix formed from the Information matrix I.

Definition 3.6. A *pure sign matrix* is a sign matrix in which each element is a sign symbol, i.e. +, 0, or -.

The importance of the Information matrix is that it contains the elements that will be used in forming sign tables. If we convert the Information matrix of a model into a pure sign matrix by restricting the system space, as explained in Section 3.1, then the sign tables of the model will be pure sign tables.

Now let us give an example on how to perform sign analysis on a simple system. Suppose we have a system which can be modeled as follows:

Equations

$F_1=z_1^2-z_2=0$

$F_2=z_2+z_3=0$

Parameters

$z_1,\ z_2,\ z_3$

The partial derivatives are

$$I = \begin{array}{c} \\ F_1 \\ F_2 \end{array} \begin{array}{ccc} z_1 & z_2 & z_3 \\ \end{array}$$

$$I = \begin{array}{c|ccc} & z_1 & z_2 & z_3 \\ \hline F_1 & 2z_1 & -1 & 0 \\ F_2 & 0 & 1 & 1 \end{array}$$

and the I_s matrix is

$$I_s = \begin{array}{c|ccc} & z_1 & z_2 & z_3 \\ \hline F_1 & [z_1] & - & 0 \\ F_2 & 0 & + & + \end{array}$$

Since both F_1 and F_2 are functions of two parameters, we use Table 2, which is the general form of the sign table for a two-parameter function. Substituting the signs of partial derivatives into the table, we obtain Tables 6 and 7 as the sign tables for F_1 and F_2, respectively.

Table 6 contains elements that are expressed in terms of the sign of z_1. We must know whether $z_1>0$, $z_1=0$, or $z_1<0$ in order to evaluate $[z_1]$ as +, 0, or −, respectively.

Suppose that, for the purpose of the analysis, we let $z_1>0$. Then Table 6 results in Table 8.

TABLE 6. Sign table of $F=z_1^2-z_2=0$

$[dz_1]$	$[dz_2]$
$[z_1]$	+
$[\overline{z_1}]$	−
0	0

TABLE 7. Sign table of $F=z_2+z_3=0$

$[dz_2]$	$[dz_3]$
+	−
−	+
0	0

TABLE 8. Pure sign table of $F=z_1^2-z_2=0$ for $[z_1]=+$

$[dz_1]$	$[dz_2]$
+	+
−	−
0	0

Now using the pure sign tables shown in Tables 8 and 7 of functions F_1 and F_2, respectively, we answer the following questions. Let T_1 represent table of F_1 and T_2 represent table of F_2. θ denotes the perturbation formula. Keep in mind that the following results are valid only for the region of the system space where z_1 is positive.

Example 3.2.1. How z_2 and z_3 change if we increase z_1?

$\theta=(\{z_1\}=+)$

$T_1 \underset{\theta}{\bowtie} T_2$ is shown in Table 9. There is one row (one possibility), so we conclude that

z_1 being positive if z_1 is increased then z_2 increases and z_3 decreases.

Example 3.2.2. How z_1 and z_3 change if we decrease z_2?

$\theta=(\{z_2\}=-)$

$T_1 \underset{\theta}{\bowtie} T_2$ is shown in Table 10. The result is

z_1 being positive if z_2 is decreased then z_1 decreases and z_3 increases.

Example 3.2.3. How z_3 changes if we increase z_1 and decrease z_2?

$\theta=(\{z_1\}=+\wedge\{z_2\}=-)$

The operation $T_1 \underset{\theta}{\bowtie} T_2$ leads to an empty table so we conclude

z_1 being positive it is not possible to increase z_1 and decrease z_2 at the same time.

TABLE 9. Result of Example 3.2.1 for $z_1>0$

$[dz_1]$	$[dz_2]$	$[dz_3]$
+	+	−

TABLE 10. Result of Example 3.2.2 for $z_1>0$

$[dz_1]$	$[dz_2]$	$[dz_3]$
−	−	+

Let us also consider the case when $z_1<0$. In this case, Table 6 results in Table 11. Using Tables 11 and 7 as T_1 and T_2, respectively, we answer the above questions which result in Tables 12, 13, and 14. Thus we conclude that

Example 3.2.1. *z_1 being negative, if z_1 is increased then z_2 decreases and z_3 increases.*

Example 3.2.2. *z_1 being negative, if z_2 is decreased then both z_1 and z_3 increase.*

Example 3.2.3. *z_1 being negative, if z_1 is increased and z_2 is decreased then z_3 increases.*

TABLE 11. Pure sign table of $F=z_1^2-z_2=0$ for $[z_1]=-$

$[dz_1]$	$[dz_2]$
−	+
+	−
0	0

TABLE 12. Result of Example 3.2.1 for $z_1<0$

$[dz_1]$	$[dz_2]$	$[dz_3]$
+	−	+

TABLE 13. Result of Example 3.2.2 for $z_1<0$

$[dz_1]$	$[dz_2]$	$[dz_3]$
+	−	+

TABLE 14. Result of Example 3.2.3 for $z_1<0$

$[dz_1]$	$[dz_2]$	$[dz_3]$
+	−	+

The two sets of results for the cases $z_1>0$ and $z_1<0$ can be merged, if desired, to yield the following:

Example 3.2.1. *If z_1 is increased then z_2 increases and z_3 decreases if z_1 is positive, z_2 decreases and z_3 increases if z_1 is negative.*

Example 3.2.2. *If z_2 is decreased then z_3 increases and z_1 decreases if z_1 is positive, z_1 increases if z_1 is negative.*

Example 3.2.3. *We can increase z_1 and decrease z_2 only if z_1 is negative, in which case z_3 increases.*

Although all of the above questions resulted in single-row tables (Tables 9, 10, 12, 13, and 14), in general there may be several possibilities. The number of possibilities is determined in part by the number of conjuncts in the perturbation formula - the more the number of conjuncts, the less the number of possibilities. This is obvious if we consider that the number of rows removed from a table during the join operation increases if the number of conjuncts increases, thus resulting into a smaller table. For example, if the perturbation formula is empty, the result will contain all the possible directions of change for all the parameters, that is, a summary of the possible *behaviors* of the system. Table 15 shows such an analysis on the previous example for the case $z_1>0$. We see from the table that this particular system can change its state in only one of two directions (excluding the case in which all parameters remain in the same state; this possibility, of course, is valid for all systems).

Note that, in the above analysis, we did not distinguish between the types of parameters nor between the types of variables. If this classification is to be done prior to the analysis, an analyst will normally be interested in finding the signs of the endogenous variables for a given set of signs of the exogenous variables and constants. In this case, the analysis will usually result in a few number of possibilities.

Sign analysis is completely a qualitative technique. It does not require to know the quantitative aspects of the model. In other words, the values of the parameters are immaterial in the analysis. Actual values of the parameters in the equations can remain unspecified because all that

TABLE 15. Result of $\in$-join for $z_1>0$

$[dz_1]$	$[dz_2]$	$[dz_3]$
+	+	−
−	−	+
0	0	0

matters is which parameters appear in which equations with nonzero coefficients. That is, sign analysis is performed by using the directions of change of the parameters, not by using their values. (Here we must note that the restrictions we impose on the system space that are necessary to evaluate the signs of partial derivatives do not interfere with the qualitativeness of the technique).

As we give the model of a system and indicate a perturbation on the model, the analysis may or may not result in a unique combination of signs for the parameters (i.e. a single row in the resulting table). We may be encountered with a table containing more than one possibility. This is basically due to the fact that sign analysis depends entirely on qualitative information. However, this situation should not be regarded as a disadvantage of the technique. This simply means that the system behaves differently in different states, since sign tables generate all the possible solutions for the remaining parameters regardless of the values of the given parameters. This situation simply is a property of the technique and should be perceived as an advantage, rather than a disadvantage, of the technique over quantitative techniques. Why it is an advantage becomes clear if we consider that the analysis allows us to see all the solutions that are possible for the given system without forcing us to choose a particular system state. This is not possible to achieve using quantitative techniques unless we exhaustively explore all numerical combinations of values of the parameters of the system and all possible amounts of the changes of the parameters in the perturbation.

Definition 3.7. An *actual solution* for a model is a solution (a combination of directions of change of parameters) which is indeed a solution for the system the model represents.

Definition 3.8. A *spurious solution* for a sign analysis of a model is a solution that participates in the sign table resulting from the analysis but is not an actual solution for the model.

Using the above definitions we will prove two important aspects of the sign analysis technique in the following propositions.

Proposition 3.2. Suppose that we have a model of n functions $F_1,\ldots,F_n$. Let $T_1,\ldots,T_n$ denote the pure sign tables of these functions $F_1,\ldots,F_n$, respectively. Let θ be a perturbation formula for the model. Call the table resulting from the operation $T_1 \bowtie_{\theta} T_2 \bowtie_{\theta} \ldots \bowtie_{\theta} T_{n-1} \bowtie_{\theta} T_n$, T. (That is, T is the table resulting from performing sign analysis indicated by the formula θ on the model). Then T contains *all* the actual solutions of the analysis.

Proof. The proof follows directly from Proposition 3.1. According to Proposition 3.1, $T_1,\ldots,T_n$ include all the actual directions of change of the parameters appearing in their respective equations. Since T is obtained by performing a join operation on $T_1,\ldots,T_n$ wrt θ, we must examine this operation. In step 2 of the definition of join (Definition 3.3), the rows that contain different directions of change for the same parameter are excluded from the cartesian product of the tables. These rows indeed cannot participate in an actual solution since a parameter cannot change in different directions at the same time. In step 4, the rows which do not agree with θ are removed. These rows also cannot participate in an actual solution since they are inconsistent with the desired perturbation. The join operation does not remove any other rows, thus it does not prevent any actual solution from taking place in T. Therefore we conclude that T includes all the actual solutions. □

Proposition 3.3. Suppose that we have a model of n functions $F_1,\ldots,F_n$. Let $T_1,\ldots,T_n$ denote the pure sign tables of these functions $F_1,\ldots,F_n$, respectively. Let θ be a perturbation formula for the model. Call the table resulting from the operation $T_1 \bowtie_\theta T_2 \bowtie_\theta \ldots \bowtie_\theta T_{n-1} \bowtie_\theta T_n$, T. (That is, T is the table resulting from performing sign analysis indicated by the formula θ on the model). Then T may include spurious solutions.

Proof. We will prove the proposition by giving an example without resorting to details. Consider a system modeled by the following two functions

$$F_1 = z_1 - z_2 - z_3 = 0$$

$$F_2 = 3z_1 - 2z_2 - 7 = 0$$

The I_s matrix is

$$I_s = \begin{array}{c} \\ F_1 \\ F_2 \end{array} \begin{array}{c} \begin{array}{ccc} z_1 & z_2 & z_3 \end{array} \\ \left[\begin{array}{ccc} + & - & - \\ + & - & 0 \end{array} \right] \end{array}$$

The sign tables of these functions, say T_1 and T_2, are shown in Tables 16 and 17. We wish to analyse the directions of change of z_1 and z_2 in case z_3 is increased. To answer this question, we perform the operation $T_1 \rhd\!\lhd_{\theta} T_2$ where $\theta=(\{z_3\}=+)$. The resulting table is shown in Table 18. We see that there are two solutions which can be stated as follows:

If z_3 is increased then either both z_1 and z_2 increase or both z_1 and z_2 decrease.

However, only one of these solutions is actual: z_1 and z_2 both decrease. The other one is a spurious solution. To see this, suppose that v_1, v_2, and v_3 constitute a particular state of the system, i.e. values of z_1, z_2, and z_3, respectively. Now suppose that all three parameters are increased. Let i_1, i_2, and i_3 stand for the amount of increases in z_1, z_2, and z_3, respectively, ($i_1>0$, $i_2>0$, $i_3>0$). Then for the equations to hold, we have

$$F_1=(v_1+i_1)-(v_2+i_2)-(v_3+i_3)=0$$
$$F_2=3(v_1+i_1)-2(v_2+i_2)-7 \qquad =0$$

After rearranging the terms, we get

$$F_1=v_1-v_2-v_3+i_1-i_2-i_3=0$$
$$F_2=3v_1-2v_2-7+3i_1-2i_2=0$$

TABLE 16. Sign table of $F=z_1-z_2-z_3=0$

$[dz_1]$	$[dz_2]$	$[dz_3]$
+	−	+
+	+	−
+	+	+
+	+	0
+	0	+
−	−	−
−	−	+
−	−	0
−	+	−
−	0	−
0	−	+
0	+	−
0	0	0

TABLE 17. Sign table of $F=3z_1-2z_2-7=0$

$[dz_1]$	$[dz_2]$
+	+
−	−
0	0

TABLE 18. Result of example in Proposition 3.3

$[dz_1]$	$[dz_2]$	$[dz_3]$
+	+	+
−	−	+

Since the values v_1, v_2, and v_3 constitute a state of the system , we have

$$v_1-v_2-v_3=0$$
$$3v_1-2v_2-7=0$$

Substituting these equalities, we find that

$$i_1-i_2-i_3=0$$
$$3i_1-2i_2=0$$

Written explicitly for i_1,

$$i_1=i_2+i_3$$
$$i_1=\frac{2}{3}i_2$$

The first equality says that the increase in z_1 is the sum of the increases in z_2 and z_3, i.e. the increase in z_1 must be greater than the increase in z_2 since $i_3>0$. On the other hand, the second equality says that the increase in z_1 must be less than the increase in z_2, which leads to a contradiction. Thus the solution where z_1 and z_2 both increase as a result of an increase in z_3 does not represent an actual solution although it takes place in Table 18. Therefore we conclude that a sign table resulting from sign analysis may contain spurious solutions. □

The main reason of getting spurious solutions is that, as can be seen from the example in the proof, sign tables do not contain any information about the amounts of change of the parameters. That is, we do not know how much a parameter increases or how much it decreases. All we know is that it increases or decreases. This is simply because we do not exploit the quantitative information that is available implicitly in the modeling equations, but we only use the qualitative part of this information. For example, what causes one of the solutions in the above example to be spurious are the coefficients of z_1 and z_2 (namely, 3 and 2) of the second equation. If these two coefficients were interchanged, then that solution would be actual.

This situation results from the ambiguity inherent in qualitative techniques, as discussed in [2,3,4,5,7]. However, we emphasize that the quantitative information that must be used to prevent spurious solutions is already present in the model, and we believe that it is possible (and not so difficult) to make use of this implicit information in this direction. This is an area left for future work.

In [5], a solution that is actual but cannot be found by the technique is called a *false negative*, and a solution that is not actual but is found by the technique (i.e. a spurious solution) is called a *false positive*. As indicated by Propositions 3.2 and 3.3, there can be no false negatives but there can be false positives in sign analysis.

Before closing this section, we want to point out another important property of sign analysis.

Definition 3.9. The *inverse of a row* of a pure sign table is obtained by replacing each element of the row (which is a sign) by its inverse.

Definition 3.10. The *inverse of a pure sign table* T is a pure sign table obtained by replacing each row of T by its inverse.

Definition 3.11. The *inverse of a perturbation formula* θ is a perturbation formula obtained by replacing each direction of change symbol in θ by its inverse. That is, the inverse of $\{r_1\}=s_1\wedge\ldots\wedge\{r_k\}=s_k$, where $r_1,\ldots,r_k$ denote parameters, is $\{r_1\}=\bar{s}_1\wedge\ldots\wedge\{r_k\}=\bar{s}_k$.

Proposition 3.4. Suppose that we have a model of n functions $F_1,\ldots,F_n$. Let $T_1,\ldots,T_n$ denote the pure sign tables of these functions $F_1,\ldots,F_n$, respectively. Let θ be a perturbation formula for the model. Call the table resulting from the operation $T_1\underset{\theta}{\rhd\!\lhd}T_2\underset{\theta}{\rhd\!\lhd}\ldots\underset{\theta}{\rhd\!\lhd}T_{n-1}\underset{\theta}{\rhd\!\lhd}T_n$, T. (That is, T is the table resulting from performing sign analysis indicated by the formula θ on the model). Let $\acute{\theta}$ be the inverse of θ. Then the result of the operation $T_1\underset{\acute{\theta}}{\rhd\!\lhd}T_2\underset{\acute{\theta}}{\rhd\!\lhd}\ldots\underset{\acute{\theta}}{\rhd\!\lhd}T_{n-1}\underset{\acute{\theta}}{\rhd\!\lhd}T_n$ is the inverse of T.

Proof In the first part of the proof we will show, by referring to the proof of Proposition 3.1, that each row in a sign table of a function has an inverse in the table.

Suppose we have a function F of n parameters $z_1,\ldots,z_n$

$$F(z_1,\ldots,z_n)=0$$

The total differential of F is

$$dF=\frac{\partial F}{\partial z_1}\,dz_1+\ldots+\frac{\partial F}{\partial z_n}\,dz_n=0$$

Let T_F denote the sign table of F. Let us represent a row of the table made up of signs $s_1,\dots,s_n$ for $[dz_1],\dots,[dz_n]$, respectively, by the notation $<s_1,\dots,s_n>$.

Suppose the contrary of the above claim: T_F contains a row $<s_1,\dots,s_n>$, but does not contain a row $<\bar{s}_1,\dots,\bar{s}_n>$. T_F containing a row $<s_1,\dots,s_n>$ means that either all $s_i=0$, $i=1,\dots,n$, or there are at least two signs in the row, say s_i and s_j, $i\neq j$, such that $[\frac{\partial F}{\partial z_i} dz_i]=+$ and $[\frac{\partial F}{\partial z_j} dz_j]=-$, i.e. the change in z_i causes F to increase and the change in z_j causes F to decrease (from Proposition 3.1). In the first case, $<s_1,\dots,s_n>=<\underbrace{0,\dots,0}_{n\ times}>$, so $<\bar{s}_1,\dots,\bar{s}_n>=<\underbrace{0,\dots,0}_{n\ times}>$, i.e. the inverse of the row is itself, already in the table. For the second case, consider the inverses of s_i and s_j, $\bar{s}_i$ and $\bar{s}_j$. Then $[\frac{\partial F}{\partial z_i} dz_i]=-$ and $[\frac{\partial F}{\partial z_j} dz_j]=+$. This means that the change in z_i causes F to decrease and the change in z_j causes F to increase. Since we have at least one parameter causing F to increase and at least one parameter causing F to decrease, this row must participate in the table regardless of the signs of other parameters (from Proposition 3.1). According to this, for the signs of other parameters, s_k, $k=1,\dots,n$, $k\neq i$, $k\neq j$, in this row, let us use the inverses, i.e. $\bar{s}_k$. Then all the signs in this row are the inverses of $s_1,\dots,s_n$. So, $<\bar{s}_1,\dots,\bar{s}_n>$ is a row of T_F.

Now having proved that each row of the table T_i, $i=1,\dots,n$, has an inverse in T_i, we will show that the operation $T'_1 \bowtie_{\in} T'_2$, (i.e. the first three steps of the join operation $T'_1 \bowtie_{\theta} T'_2$), for any two tables T'_1 and T'_2, preserves this property. The first step is the cartesian product of the two tables, say T'. If T'_1 has a row $<s_1,\dots,s_k>$ and T'_2 has a row $<t_1,\dots,t_m>$ then T' will contain the row $<s_1,\dots,s_k,t_1,\dots,t_m>$. Since T'_1 must also contain the row $<\bar{s}_1,\dots,\bar{s}_k>$ and T'_2 must contain the row $<\bar{t}_1,\dots,\bar{t}_m>$ (by the above discussion), then T' will also contain $<\bar{s}_1,\dots,\bar{s}_k,\bar{t}_1,\dots,\bar{t}_m>$. Thus the first step preserves the property. In the second step, if we remove a row $<s_1,\dots,s_k>$ from T' because s_i and s_j, $i,j=1,\dots,k$, are the signs for the same parameter such that $s_i \neq s_j$, we also remove the row $<\bar{s}_1,\dots,\bar{s}_k>$ since $\bar{s}_i \neq \bar{s}_j$. Thus the second step also preserves the property. Call T'' the table resulting after the application of the second step. In the third step, if we project out the i^{th} column of T'', then any two inverse rows $<s_1,\dots,s_k>$ and $<\bar{s}_1,\dots,\bar{s}_k>$ will result in $<s_1,\dots,s_{i-1},s_{i+1},\dots,s_k>$ and $<\bar{s}_1,\dots,\bar{s}_{i-1},\bar{s}_{i+1},\dots,\bar{s}_k>$, $i=1,\dots,k$, which are still inverses. So, after $T'_1 \bowtie_{\in} T'_2$, each row will have an inverse.

Suppose that we have computed $T_1 \bowtie_{\in} T_2 \bowtie_{\in} \dots \bowtie_{\in} T_{n-1} \bowtie_{\in} T_n$, call this table T'. Let us apply the fourth step to T' using θ and $\acute{\theta}$, call the resulting tables T'' and T''', respectively. Let $\theta=(\{z_1\}=s_1 \wedge \dots \wedge \{z_k\}=s_k)$ where $z_1,\dots,z_k$ are some of the parameters of the model. Suppose there is a row $<t_1,\dots,t_m>$ in T' which agrees with θ, i.e. $s_1=t_{i_1},\dots,s_k=t_{i_k}$, where t_{i_j} is the sign for $[dz_j]$, $j=1,\dots,k$. Then this row will participate in T''. T' must also contain the row $<\bar{t}_1,\ldots,\bar{t}_m>$ and this row agrees with θ. To see this, note that $\theta=(\{z_1\}=\bar{s}_1\wedge\ldots\wedge\{z_k\}=\bar{s}_k)$ and $\bar{s}_1=\bar{t}_{i_1},\ldots,\bar{s}_k=\bar{t}_{i_k}$, $j=1,\ldots,k$. Thus this row will participate in T'''. Therefore T''' is the inverse of T''.

It remains to show that, for any perturbation formula α, $T_1\bowtie_{\alpha}T_2\bowtie_{\alpha}\ldots\bowtie_{\alpha}T_{n-1}\bowtie_{\alpha}T_n$ results in the same table as applying the fourth step wrt α to the table $T_1\bowtie_{\epsilon}T_2\bowtie_{\epsilon}\ldots\bowtie_{\epsilon}T_{n-1}\bowtie_{\epsilon}T_n$. But this is intuitive since the only difference between the two operations is that in the latter case the rows that are inconsistent with α are removed at the end instead of during each operation.

Therefore we conclude that $T_1\bowtie_{\theta}T_2\bowtie_{\theta}\ldots\bowtie_{\theta}T_{n-1}\bowtie_{\theta}T_n$ is the inverse of $T_1\bowtie_{\theta}T_2\bowtie_{\theta}\ldots\bowtie_{\theta}T_{n-1}\bowtie_{\theta}T_n$. □

3.3. *Sign Analysis of Gains*

We have explained sign analysis technique in detail in Section 3.2. In the present section we will discuss the application of the technique to gains.

Consider a set of functions

$$\begin{array}{l} F_1(x_1,\ldots,x_n,y_1,\ldots,y_m)=0 \\ \vdots \\ F_n(x_1,\ldots,x_n,y_1,\ldots,y_m)=0 \end{array} \qquad (3.3)$$

where $x_1,\ldots,x_n$ represent endogenous variables and $y_1,\ldots,y_m$ represent exogenous variables and constants. Suppose that we change the value of one of the exogenous variables, say y_j, $j=1,\dots,m$, while keeping the values of other exogenous variables y_i, $i=1,\dots,m$ and $i\neq j$, unchanged. As will be derived in Section 4.1, we can write

$$\begin{array}{l}\frac{\partial F_1}{\partial y_j}\frac{\partial y_j}{\partial y_j}+\frac{\partial F_1}{\partial x_1}\frac{\partial x_1}{\partial y_j}+\dots+\frac{\partial F_1}{\partial x_n}\frac{\partial x_n}{\partial y_j}=0\\ \vdots \\ \frac{\partial F_n}{\partial y_j}\frac{\partial y_j}{\partial y_j}+\frac{\partial F_n}{\partial x_1}\frac{\partial x_1}{\partial y_j}+\dots+\frac{\partial F_n}{\partial x_n}\frac{\partial x_n}{\partial y_j}=0\end{array} \tag{3.4}$$

The partial derivative $\frac{\partial x_i}{\partial y_j}$ represents the ratio of the change to be occurred in the value of the endogenous variable x_i for a change specified in the value of the exogenous variable/constant y_j and it is called the *gain of x_i with respect to y_j* in the control literature and the *marginal value of x_i with respect to y_j* in the economics literature. We will prefer to use the term gain because we will be mostly interested in physical systems.

For each of the equations in (3.4) we form a sign table. In this case the columns of the sign tables will be titled as $[\frac{\partial x_i}{\partial y_j}]$, $i=1,\dots,n$, and $[\frac{\partial y_j}{\partial y_j}]$. That is, the columns represent signs of gains. We note that $\frac{\partial y_j}{\partial y_j}$ is unity, thus $[\frac{\partial y_j}{\partial y_j}]=+$. Therefore we will only consider those rows of the tables in which this condition holds, i.e. a subset of the tables will be used. (Of course, this is applicable to tables whose functions include the parameter y_j).

The sign of a gain, $[\frac{\partial x_i}{\partial y_j}]$, represents the direction of change in the value of the endogenous variable x_i for a change in the value of the exogenous variable y_j. $[\frac{\partial x_i}{\partial y_j}]=+$ means that x_i and y_j change in the same direction. That is, increasing y_j causes x_i to increase and decreasing y_j causes x_i to decrease. $[\frac{\partial x_i}{\partial y_j}]=-$ means that x_i and y_j change in opposite directions. That is, increasing y_j causes x_i to decrease and decreasing y_j causes x_i to increase. $[\frac{\partial x_i}{\partial y_j}]=0$ means that the change in y_j has no effect on x_i.

When the sign tables are formed, converted to pure sign tables, and the rows in which $[\frac{\partial y_j}{\partial y_j}]\neq+$ are eliminated, using these reduced pure sign tables (call these $T_1,\ldots,T_n$), we compute $T_1 \underset{\in}{\rhd\!\!=\!\!\lhd} T_2 \underset{\in}{\rhd\!\!=\!\!\lhd} \ldots \underset{\in}{\rhd\!\!=\!\!\lhd} T_{n-1} \underset{\in}{\rhd\!\!=\!\!\lhd} T_n$. This shows us in which direction the endogenous variables $x_1,\ldots,x_n$ change for a change in the exogenous variable y_j.

As an example, consider the model of Section 3.2,

Equations

$F_1=z_1^2-z_2=0$

$F_2=z_2+z_3=0$

Parameters

z_1 (exo)

z_2, z_3 (endo)

For the sake of illustration, we have made the types of the parameters explicit (This was not necessary for the sign analysis of total differentials). Converting the equations into the form of (3.4) for the exogenous variable z_1, we get

$$\frac{\partial F_1}{\partial z_1}\frac{\partial z_1}{\partial z_1}+\frac{\partial F_1}{\partial z_2}\frac{\partial z_2}{\partial z_1}+\frac{\partial F_1}{\partial z_3}\frac{\partial z_3}{\partial z_1}=0$$
$$\frac{\partial F_2}{\partial z_1}\frac{\partial z_1}{\partial z_1}+\frac{\partial F_2}{\partial z_2}\frac{\partial z_2}{\partial z_1}+\frac{\partial F_2}{\partial z_3}\frac{\partial z_3}{\partial z_1}=0 \qquad (3.5)$$

The partial derivatives are shown in the Information matrix,

$$I = \begin{array}{c|ccc} & z_1 & z_2 & z_3 \\ F_1 & 2z_1 & -1 & 0 \\ F_2 & 0 & 1 & 1 \end{array}$$

Substituting into (3.5), we get

$$2z_1\frac{\partial z_1}{\partial z_1}-\frac{\partial z_2}{\partial z_1}=0$$
$$\frac{\partial z_2}{\partial z_1}+\frac{\partial z_3}{\partial z_1}=0 \qquad (3.6)$$

Assuming $z_1>0$, the I_s matrix evaluates to

$$I_s = \begin{array}{c|ccc} & z_1 & z_2 & z_3 \\ F_1 & + & - & 0 \\ F_2 & 0 & + & + \end{array} \qquad (3.7)$$

Now we form the pure sign tables of the equations shown in (3.6) using the signs of partial derivatives (3.7). The tables are shown in Tables 19 and 20. Note that the columns are titled with the signs of gains. We also note that $[\frac{\partial z_1}{\partial z_1}]$ must be +, so only the first row of Table 19 is valid. Therefore Table 19 reduces to Table 21. The ∈-join of Tables 21 and 20 results in Table 22, which leads us to the following result:

TABLE 19. Sign table of gains of $F=z_1^2-z_2=0$ wrt z_1

$[\frac{\partial z_1}{\partial z_1}]$	$[\frac{\partial z_2}{\partial z_1}]$
+	+
−	−
0	0

TABLE 20. Sign table of gains of $F=z_2+z_3=0$ wrt z_1

$[\frac{\partial z_2}{\partial z_1}]$	$[\frac{\partial z_3}{\partial z_1}]$
+	−
−	+
0	0

TABLE 21. Reduced form of Table 19

$[\frac{\partial z_1}{\partial z_1}]$	$[\frac{\partial z_2}{\partial z_1}]$
+	+

TABLE 22. Result of sign analysis for gains

$[\frac{\partial z_1}{\partial z_1}]$	$[\frac{\partial z_2}{\partial z_1}]$	$[\frac{\partial z_3}{\partial z_1}]$
+	+	−

z_1 being positive, when we change the value of z_1, z_2 changes in the same direction as z_1 and z_3 changes in the opposite direction of z_1.

Let us examine the relationship between sign analysis of gains and sign analysis of total differentials. In Section 3.2, we asked the question

How z_2 and z_3 change if we increase z_1?

on the same model for the case $z_1>0$, and arrived at the answer

z_1 being positive, if z_1 is increased then z_2 increases and z_3 decreases.

From Proposition 3.4, we also know that

z_1 being positive, if z_1 is decreased then z_2 decreases and z_3 increases.

The combination of these two clauses is identical to the one extracted from Table 22.

It must not be surprising that sign analysis of gains and sign analysis of total differentials yield identical results. To see this, let us write the total differentials of the functions in (3.3)

$$\begin{aligned} dF_1 &= \frac{\partial F_1}{\partial x_1}\,dx_1+\ldots+\frac{\partial F_1}{\partial x_n}\,dx_n+\frac{\partial F_1}{\partial y_1}\,dy_1+\ldots+\frac{\partial F_1}{\partial y_m}\,dy_m=0 \\ &\vdots \\ dF_n &= \frac{\partial F_n}{\partial x_1}\,dx_1+\ldots+\frac{\partial F_n}{\partial x_n}\,dx_n+\frac{\partial F_n}{\partial y_1}\,dy_1+\ldots+\frac{\partial F_n}{\partial y_m}\,dy_m=0 \end{aligned} \tag{3.8}$$

As we did above, suppose that we change the value of y_j, $j=1,\ldots,m$, while keeping the values of y_i, $i=1,\ldots,m$ and $i\neq j$, unchanged. So $dy_i=0$ and (3.8) becomes

$$\begin{aligned} dF_1 &= \frac{\partial F_1}{\partial y_j}\,dy_j+\frac{\partial F_1}{\partial x_1}\,dx_1+\ldots+\frac{\partial F_1}{\partial x_n}\,dx_n=0 \\ &\vdots \\ dF_n &= \frac{\partial F_n}{\partial y_j}\,dy_j+\frac{\partial F_n}{\partial x_1}\,dx_1+\ldots+\frac{\partial F_n}{\partial x_n}\,dx_n=0 \end{aligned} \tag{3.9}$$

Now we form the sign tables of functions in (3.9). The columns will be titled as $[dx_i]$, $i=1,\ldots,n$, and $[dy_j]$. The point is that the sign tables for gains in (3.4) and the sign tables for total differentials in (3.9) will be exactly the same since the equations have the same coefficients. The only difference is in the titles of the columns of the sign tables. For instance, compare Tables 8 and 7 with Tables 19 and 20.

In sign analysis of gains we require $[\frac{\partial y_j}{\partial y_j}]=+$ and reduce the tables. On the other hand, in sign analysis of total differentials, we perform the same analysis first for $[dy_j]=+$ then for $[dy_j]=-$, and combine the results (or, for either $[dy_j]=+$ or $[dy_j]=-$, and then use Proposition 3.4).

In the method of sign analysis of gains as described above, we can change only one exogenous variable. Now let us expand the method. If we remove the restriction on (3.4), i.e. if all the exogenous variables are allowed to be changed, it reduces to the following equations in terms of signs (the use of y_j is arbitrary here, we could use any other exogenous variable as well)

$$\begin{array}{l}
[\frac{\partial F_1}{\partial x_1}\frac{\partial x_1}{\partial y_j}]+\ldots+[\frac{\partial F_1}{\partial x_n}\frac{\partial x_n}{\partial y_j}]+[\frac{\partial F_1}{\partial y_1}\frac{\partial y_1}{\partial y_j}]+\ldots+[\frac{\partial F_1}{\partial y_m}\frac{\partial y_m}{\partial y_j}]=0 \\
\vdots \\
[\frac{\partial F_n}{\partial x_1}\frac{\partial x_1}{\partial y_j}]+\ldots+[\frac{\partial F_n}{\partial x_n}\frac{\partial x_n}{\partial y_j}]+[\frac{\partial F_n}{\partial y_1}\frac{\partial y_1}{\partial y_j}]+\ldots+[\frac{\partial F_n}{\partial y_m}\frac{\partial y_m}{\partial y_j}]=0
\end{array} \qquad (3.10)$$

As before we form the sign tables of these equations. The columns will be titled as $[\frac{\partial x_i}{\partial y_j}]$, i=1,...,n, and $[\frac{\partial y_k}{\partial y_j}]$, k=1,...,m. We again require $[\frac{\partial y_j}{\partial y_j}]$ =+, so remove rows which disagree with this condition. What this analysis is about can be summarized as follows:

We change y_j. We can also change other exogenous variables. In which direction wrt y_j the endogenous variables change?

Note that $[\frac{\partial y_i}{\partial y_j}]$, i≠j, does not correspond to the phrase

In which direction y_i changes if we change y_j?

instead it corresponds to the phrase

In which direction, wrt y_j, do we change y_i?

because both are exogenous variables and a change in y_j does not affect y_i, i.e. we can change y_i independent of y_j[3]. $[\frac{\partial y_i}{\partial y_j}]$ =+ means that we change y_i in the same direction as y_j, $[\frac{\partial y_i}{\partial y_j}]$ =- means that we change y_i in the opposite direction of y_j, and $[\frac{\partial y_i}{\partial y_j}]$ =0 means that we do not change y_i.

[3]Of course, if we do not impose any distinction between the types of the parameters, then we can write (3.10) for any parameter z_j and perform the analysis for any $[\frac{\partial z_i}{\partial z_j}]$.

As an example, consider the following model

Equations

$F_1 = y_1^2 - x_1 = 0$

$F_2 = x_1 + x_2 + y_2 = 0$

Parameters

x_1, x_2 (endo)

y_1, y_2 (exo)

Suppose that we ask the following question:

We change y_1 and y_2 in opposite directions. How do x_1 and x_2 change?

Using (3.10), we write

$$\begin{aligned}
&\left[\frac{\partial F_1}{\partial x_1}\frac{\partial x_1}{\partial y_1}\right]+\left[\frac{\partial F_1}{\partial x_2}\frac{\partial x_2}{\partial y_1}\right]+\left[\frac{\partial F_1}{\partial y_1}\frac{\partial y_1}{\partial y_1}\right]+\left[\frac{\partial F_1}{\partial y_2}\frac{\partial y_2}{\partial y_1}\right]=0\\
&\left[\frac{\partial F_2}{\partial x_1}\frac{\partial x_1}{\partial y_1}\right]+\left[\frac{\partial F_2}{\partial x_2}\frac{\partial x_2}{\partial y_1}\right]+\left[\frac{\partial F_2}{\partial y_1}\frac{\partial y_1}{\partial y_1}\right]+\left[\frac{\partial F_2}{\partial y_2}\frac{\partial y_2}{\partial y_1}\right]=0
\end{aligned} \qquad (3.11)$$

The I matrix is

$$I = \begin{array}{c|cccc} & x_1 & x_2 & y_1 & y_2 \\ \hline F_1 & -1 & 0 & 2y_1 & 0 \\ F_2 & 1 & 1 & 0 & 1 \end{array}$$

Substituting into (3.11), we get

$$\begin{aligned}
&\left[-\frac{\partial x_1}{\partial y_1}\right]+\left[2y_1\frac{\partial y_1}{\partial y_1}\right]=0\\
&\left[\frac{\partial x_1}{\partial y_1}\right]+\left[\frac{\partial x_2}{\partial y_1}\right]+\left[\frac{\partial y_2}{\partial y_1}\right]=0
\end{aligned} \qquad (3.12)$$

Assuming $y_1>0$, the I_s matrix evaluates to

$$I_s = \begin{array}{c} \\ F_1 \\ F_2 \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & x_2 & y_1 & y_2 \end{array} \\ \left[\begin{array}{cccc} - & 0 & + & 0 \\ + & + & 0 & + \end{array}\right] \end{array} \qquad (3.13)$$

The sign tables of the equations in (3.12) using (3.13) are shown in Tables 23 and 24. Since $[\frac{\partial y_1}{\partial y_1}]=+$, Table 23 reduces to Table 25. We set $[\frac{\partial y_2}{\partial y_1}]=-$ from the question. Thus $\theta=([\frac{\partial y_2}{\partial y_1}]=-)$[4]. The θ-join of the two tables, Tables 25 and 24, results in Table 26.

Thus we conclude that

y_1 being positive, if y_1 and y_2 are changed in opposite directions, then x_1 changes in the same direction as y_1 and x_2 may change in any direction (wrt y_1).

Note that we could perform the same analysis wrt y_2 instead of y_1. That is, we could write (3.11) wrt y_2. Then the tables would be titled with $[\frac{\partial x_1}{\partial y_2}]$, $[\frac{\partial x_2}{\partial y_2}]$, $[\frac{\partial y_1}{\partial y_2}]$, and $[\frac{\partial y_2}{\partial y_2}]$; we would set $[\frac{\partial y_1}{\partial y_2}]=-$; and the results would indicate directions of change wrt y_2. Since y_1 and y_2 change in opposite directions, it is not hard to see that the analysis in this case would result in the following clause:

[4]In sign analysis of gains, a perturbation formula is formed from the conjunction of signs of partial derivatives instead of directions of change of parameters.

TABLE 23. Sign table of gains of $F=y_1^2-x_1=0$ wrt y_1

$[\frac{\partial x_1}{\partial y_1}]$	$[\frac{\partial y_1}{\partial y_1}]$
−	−
+	+
0	0

TABLE 24. Sign table of gains of $F=x_1+x_2+y_2=0$ wrt y_1

$[\frac{\partial x_1}{\partial y_1}]$	$[\frac{\partial x_2}{\partial y_1}]$	$[\frac{\partial y_2}{\partial y_1}]$
+	+	−
+	−	+
+	−	−
+	−	0
+	0	−
−	+	+
−	+	−
−	+	0
−	−	+
−	0	+
0	+	−
0	−	+
0	0	0

TABLE 25. Reduced form of Table 23

$[\frac{\partial x_1}{\partial y_1}]$	$[\frac{\partial y_1}{\partial y_1}]$
+	+

TABLE 26. Result of opposite changes in y_1 and y_2

$[\frac{\partial x_1}{\partial y_1}]$	$[\frac{\partial y_1}{\partial y_1}]$	$[\frac{\partial x_2}{\partial y_1}]$	$[\frac{\partial y_2}{\partial y_1}]$
+	+	+	−
+	+	−	−
+	+	0	−

y_1 being positive, if y_1 and y_2 are changed in opposite directions, then x_1 changes in the opposite direction of y_2 and x_2 may change in any direction (wrt y_2)

which is identical with the previous one.

As a final note let us point out that this extended form of sign analysis of gains is also similar to sign analysis of total differentials. The reason is, as in the previous case, that the two equation sets of (3.8) and (3.10) have the same coefficients, so produce the same sign tables. In sign analysis of gains, we change y_j and indicate $[\frac{\partial y_i}{\partial y_j}]$, $i=1,...,m$, $i \neq j$. On the other hand, in sign analysis of total differentials, we perform the sign analysis first for $[dy_j]=+$ then for $[dy_j]=-$, and combine the results. For $[dy_j]=+$, $[dy_i]=[\frac{\partial y_i}{\partial y_j}]$. For $[dy_j]=-$, $[dy_i]=[\overline{\frac{\partial y_i}{\partial y_j}}]$. Therefore the two methods will produce the same results.

IV. VALUE ANALYSIS

Another technique that will be introduced here is called value analysis. Value analysis is based on the use of gains and examines in which amounts the parameters change due to a perturbation in the system. In this respect it can be regarded as the quantitative complement of sign analysis.

The idea can be explained as follows: The model of a system is given. We specify a state of the system (i.e. values of the parameters). Then we compute the gains for the given state and examine the total differentials of the parameters.

4.1. Gains

Let us consider a model of the form of

$$\begin{aligned} &F_1(x_1,\dots,x_n,y_1,\dots,y_m)=0 \\ &\vdots \\ &F_n(x_1,\dots,x_n,y_1,\dots,y_m)=0 \end{aligned}$$

where $x_1,\dots,x_n$ represent endogenous variables and $y_1,\dots,y_m$ represent exogenous variables and constants. The number of endogenous variables is equal to the number of functions, so the degrees of freedom in the system is equal to m. Thus the system can be solved uniquely for the values of the endogenous variables for a given set of values of the exogenous variables and constants, provided that the system is nonsingular.

Suppose that we change the value of one of the exogenous variables, say y_j, $j=1,\dots,m$, while keeping the values of other exogenous variables y_i, $i=1,\dots,m$ and $i\neq j$, unchanged. We want to examine the change in the values of the endogenous variables.

The total differentials of the functions are

$$\begin{aligned} dF_1 &= \frac{\partial F_1}{\partial x_1}\,dx_1+\dots+\frac{\partial F_1}{\partial x_n}\,dx_n+\frac{\partial F_1}{\partial y_1}\,dy_1+\dots+\frac{\partial F_1}{\partial y_m}\,dy_m=0 \\ &\vdots \\ dF_n &= \frac{\partial F_n}{\partial x_1}\,dx_1+\dots+\frac{\partial F_n}{\partial x_n}\,dx_n+\frac{\partial F_n}{\partial y_1}\,dy_1+\dots+\frac{\partial F_n}{\partial y_m}\,dy_m=0 \end{aligned} \tag{4.1}$$

As we know, the total differential of a parameter z indicates the change in the value of z. Thus $dy_i=0$, $i=1,\dots,m$ and $i\neq j$, since we do not change the values of exogenous variables except y_j. So, (4.1) becomes

$$\begin{aligned} &\frac{\partial F_1}{\partial x_1}\,dx_1+\dots+\frac{\partial F_1}{\partial x_n}\,dx_n= -\frac{\partial F_1}{\partial y_j}\,dy_j \\ &\vdots \\ &\frac{\partial F_n}{\partial x_1}\,dx_1+\dots+\frac{\partial F_n}{\partial x_n}\,dx_n= -\frac{\partial F_n}{\partial y_j}\,dy_j \end{aligned} \tag{4.2}$$

This is a set of n linear equations in n unknowns $dx_1,\dots,dx_n$. Solving for unknowns and after a little bit of algebra, we get

$$
\begin{aligned}
dx_1 &= \frac{\partial x_1}{\partial y_j}\, dy_j \\
&\vdots \\
dx_n &= \frac{\partial x_n}{\partial y_j}\, dy_j
\end{aligned}
\tag{4.3}
$$

Putting these terms into (4.2) and cancelling dy_j terms, we get in scalar form

$$
\begin{aligned}
&\frac{\partial F_1}{\partial x_1}\frac{\partial x_1}{\partial y_j}+\dots+\frac{\partial F_1}{\partial x_n}\frac{\partial x_n}{\partial y_j} = -\frac{\partial F_1}{\partial y_j} \\
&\vdots \\
&\frac{\partial F_n}{\partial x_1}\frac{\partial x_1}{\partial y_j}+\dots+\frac{\partial F_n}{\partial x_n}\frac{\partial x_n}{\partial y_j} = -\frac{\partial F_n}{\partial y_j}
\end{aligned}
\tag{4.4}
$$

or in vector form

$$
\begin{bmatrix}
\frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\
\vdots & & \vdots \\
\frac{\partial F_n}{\partial x_1} & \cdots & \frac{\partial F_n}{\partial x_n}
\end{bmatrix}
\begin{bmatrix}
\frac{\partial x_1}{\partial y_j} \\
\vdots \\
\frac{\partial x_n}{\partial y_j}
\end{bmatrix}
= -
\begin{bmatrix}
\frac{\partial F_1}{\partial y_j} \\
\vdots \\
\frac{\partial F_n}{\partial y_j}
\end{bmatrix}
\tag{4.5}
$$

The above matrix is the Jacobian matrix of the system. By denoting the column vector at the left-hand side of the equality by $\mathbf{x}$, and the one at the right-hand side by $\mathbf{f}$, the notation can be simplified as

$$
J\mathbf{x} = -\mathbf{f} \tag{4.6}
$$

These equations constitute a set of simultaneous linear equations in n unknowns $\frac{\partial x_1}{\partial y_j},\dots,\frac{\partial x_n}{\partial y_j}$ and can be solved by any method used for solving simultaneous linear equations, such as Gaussian Elimination or Cramer´s method, for a given state of the system.

As explained in Section 3.3, the partial derivative $\frac{\partial x_i}{\partial y_j}$ represents the ratio of the change to be occurred in the value of the endogenous variable x_i for a sufficiently small change[5] specified in the value of the exogenous variable/ constant y_j and is called the gain of x_i wrt y_j.

The above derivation is performed for a single y_j. What we have done is that we specify a particular state of the system (i.e. the values of the parameters $x_1,\dots,x_n,y_1,\dots,y_m$) and a parameter y_j, $j=1,\dots,m$, whose value is to be changed. We then compute the nxn partial derivatives $\frac{\partial F_i}{\partial x_k}$, $i,k=1,\dots,n$, and the n partial derivatives $\frac{\partial F_i}{\partial y_j}$, $i=1,\dots,n$. Then by using (4.4), we solve for $\frac{\partial x_i}{\partial y_j}$, $i=1,\dots,n$. This gives us the gains of the endogenous variables wrt the exogenous variable y_j.

The same analysis can be performed for any exogenous variable by simply replacing the term y_j in (4.4) with the name of the desired parameter. In the notation of (4.5), the Jacobian matrix remains the same, only the elements of the vector $\underline{f}$ are to be changed. In this way we obtain all the

[5]Since the derivation involves the use of differentiation, it is only valid in the limit, i.e. for an infinitesimal change. This is due to the nonlinearity in the equations; for a linear equation we can ignore this restriction. However, as we deal with nonlinear equations in general, this point must be kept in mind when interpreting the results obtained from gains.

gains for that state of the system, $\frac{\partial x_i}{\partial y_j}$, i=1,...,n, j=1,...m.

Gains can be used to compute the total differentials of endogenous variables for a given set of values of the total differentials of exogenous variables. The total differential formula can be easily derived from (4.1) by letting $y_1,\dots,y_m$ to change (i.e. the terms $dy_1,\dots,dy_m$ do not vanish). This evaluates to

$$\begin{aligned} dx_1 &= \frac{\partial x_1}{\partial y_1} dy_1 + \dots + \frac{\partial x_1}{\partial y_m} dy_m \\ &\vdots \\ dx_n &= \frac{\partial x_n}{\partial y_1} dy_1 + \dots + \frac{\partial x_n}{\partial y_m} dy_m \end{aligned} \tag{4.7}$$

which is an extended form of (4.3) and means that the total differential of an endogenous variable is simply the sum of the contributions of each of the exogenous variables.

Now we will illustrate the point by a simple example. Consider a system modeled as follows

Equations

$F_1 = x_1 - 8y_1 + y_2 = 0$
$F_2 = -x_1 + x_2 + y_1 y_2 = 0$

Parameters

x_1, x_2 (endo)
y_1, y_2 (exo)

Suppose that the system state is given as follows[6]

$$x_1=5 \quad x_2=2 \quad y_1=1 \quad y_2=3 \tag{4.8}$$

The partial derivatives for endogenous variables are calculated as

$$\frac{\partial F_1}{\partial x_1}=1 \quad \frac{\partial F_1}{\partial x_2}=0 \quad \frac{\partial F_2}{\partial x_1}=-1 \quad \frac{\partial F_2}{\partial x_2}=1$$

So, the Jacobian matrix is

$$J = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

The partial derivatives for y_1 and y_2 are

$$\frac{\partial F_1}{\partial y_1}=-8 \quad \frac{\partial F_1}{\partial y_2}=1 \quad \frac{\partial F_2}{\partial y_1}=y_2=3 \quad \frac{\partial F_2}{\partial y_2}=y_1=1$$

that can be represented by the column vectors $\underline{f}_1$ and $\underline{f}_2$ as

$$\underline{f}_1 = \begin{bmatrix} -8 \\ 3 \end{bmatrix} \quad \underline{f}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$\underline{f}_1$ is the vector of the partial derivatives of functions for y_1 and $\underline{f}_2$ is the vector of the partial derivatives of functions for y_2. If we denote by $\underline{v}_1$ and $\underline{v}_2$ the column vectors of gains for y_1 and y_2, respectively, as in

[6]The system state can be specified either by giving the values of all the parameters as in the example or by giving the values of exogenous variables and constants from which we can solve for the values of the endogenous variables. In this second approach, a method for solving a set of simultaneous nonlinear equations must be used since the modeling equations are nonlinear in general.

$$\underline{v}_1 = \begin{bmatrix} \frac{\partial x_1}{\partial y_1} \\ \frac{\partial x_2}{\partial y_1} \end{bmatrix} \qquad \underline{v}_2 = \begin{bmatrix} \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_2} \end{bmatrix}$$

we arrive, by using (4.5), to the following two sets of simultaneous equations

$$\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial y_1} \\ \frac{\partial x_2}{\partial y_1} \end{bmatrix} = - \begin{bmatrix} -8 \\ 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_2} \end{bmatrix} = - \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

or in matrix form (by using (4.6))

$$J\underline{v}_1 = -\underline{f}_1$$

$$J\underline{v}_2 = -\underline{f}_2$$

Solving these separately we get

$$\frac{\partial x_1}{\partial y_1}=8 \quad \frac{\partial x_2}{\partial y_1}=5 \quad \frac{\partial x_1}{\partial y_2}=-1 \quad \frac{\partial x_2}{\partial y_2}=-2 \qquad (4.9)$$

The results can be interpreted as follows:

While keeping y_2 constant, an increase in the value of y_1 causes an 8 times increase in the value of x_1 and a 5 times increase in the value of x_2.

While keeping y_1 constant, an increase in the value of y_2 causes an equal amount decrease in the value of x_1 and a 2 times decrease in the value of x_2.

The interpretation in the opposite direction is also valid:

While keeping y_2 constant, a decrease in the value of y_1 causes an 8 times decrease in the value of x_1 and a 5 times decrease in the value of x_2.

While keeping y_1 constant, a decrease in the value of y_2 causes an equal amount increase in the value of x_1 and a 2 times increase in the value of x_2.

The two can be combined into a general form:

While keeping y_2 constant, a change in the value of y_1 causes an 8 times change in the value of x_1 in the same direction as y_1 and a 5 times change in the value of x_2 in the same direction as y_1.

While keeping y_1 constant, a change in the value of y_2 causes an equal amount change in the value of x_1 in the opposite direction of y_2 and a 2 times change in the value of x_2 in the opposite direction of y_2.

The results can be used to calculate the total differentials of endogenous variables. From (4.7) and (4.9), we can write

$$dx_1=\frac{\partial x_1}{\partial y_1}\,dy_1+\frac{\partial x_1}{\partial y_2}\,dy_2$$

$$dx_2=\frac{\partial x_2}{\partial y_1}\,dy_1+\frac{\partial x_2}{\partial y_2}\,dy_2$$

and

$$\begin{aligned} dx_1&=8dy_1-dy_2\\ dx_2&=5dy_1-2dy_2 \end{aligned} \tag{4.10}$$

4.2. Value Analysis of Total Differentials

Suppose we have a set of n functions

$$F_1(x_1,...,x_n,y_1,...,y_m)=0$$
$$\vdots$$
$$F_n(x_1,...,x_n,y_1,...,y_m)=0$$

where $x_1,...,x_n$ are endogenous variables and $y_1,...,y_m$ are exogenous variables and constants. Suppose also that a system state is specified. Then we can compute the gains as explained in Section 4.1 and, similar to (4.7), we can write

$$dx_1-\frac{\partial x_1}{\partial y_1}\,dy_1-...-\frac{\partial x_1}{\partial y_m}\,dy_m=0$$
$$\vdots$$
$$dx_n-\frac{\partial x_n}{\partial y_1}\,dy_1-...-\frac{\partial x_n}{\partial y_m}\,dy_m=0$$

This is a set of n simultaneous linear equations in n+m unknowns $dx_1,...,dx_n,dy_1,...,dy_m$, and can be solved by specifying the values of m of the unknowns. Normally we will be interested in solving for $dx_1,...,dx_n$ (the amounts of changes in the values of the endogenous variables) given $dy_1,...,dy_m$ (the amounts of changes in the values of the exogenous variables).

As an example, consider the model of Section 4.1, which was reproduced below

Equations

$F_1= x_1-8y_1+y_2 =0$

$F_2=-x_1+x_2+y_1 y_2=0$

Parameters

x_1, x_2 (endo)

y_1, y_2 (exo)

For the system state given in (4.8),

$$x_1=5 \quad x_2=2 \quad y_1=1 \quad y_2=3$$

the gains were computed and the total differential equations are (from (4.10)),

$$\begin{aligned} dx_1-8dy_1+ dy_2&=0 \\ dx_2-5dy_1+2dy_2&=0 \end{aligned} \tag{4.11}$$

Now we can answer questions such as the following:

Example 4.2.1. We increase y_1 and y_2 such that the increase in y_2 is twice that of y_1. How do x_1 and x_2 change?

Substituting $dy_1=1$ (a unit increase) and $dy_2=2$, we get

$dx_1=6$

$dx_2=1$

x_1 increases 6 times the increase in y_1 and x_2 increases an amount equal to the increase in y_1.

Example 4.2.2. We change y_1. We want to change y_2 in such a way that the value of x_1 does not change. How must y_2 be changed and what is the effect on x_2?

Substituting $dy_1=1$ and $dx_1=0$ into (4.11), we get

$dy_2= 8$

$dx_2=-11$

y_2 must be changed 8 times the change in y_1 and in the same direction. x_1 remains the same and x_2 changes 11 times the change in y_1 in the opposite direction.

Example 4.2.3. We want x_1 and x_2 to increase in the same amount. How must y_1 and y_2 be changed?

Substituting $dx_1=1$ and $dx_2=1$, we get

$$dy_1=\frac{1}{11}$$
$$dy_2=-\frac{3}{11}$$

We must increase y_1 $\frac{1}{11}$ times and decrease y_2 $\frac{3}{11}$ times the desired increase in x_1.

Example 4.2.4. We want a unit increase in x_1. How can we change y_1 and y_2 for this purpose and what is the effect on x_2?

Substituting $dx_1=1$, we get

$$-8dy_1 + dy_2 = -1$$
$$dx_2-5dy_1+2dy_2= 0$$

Since there remains three unknowns, the equations have infinite number of solutions. Let us perform sign analysis for this question. Without delving into details, we show the result of the sign analysis on the model for the perturbation $e=(\{x_1\}=+)$ (and using the assumptions that $y_1>0$ and $y_2>0$), in Table 27. The table can be summarized as follows:

If we decrease y_1 or keep it unchanged then we must decrease y_2 and x_2 increases. If we increase y_1 then we can change y_2 in any direction and x_2 changes in any direction.

Suppose we decided to decrease y_1 in the same amount of the increase in x_1. So, $dy_1=-1$, and we get

$$dy_2=-9$$
$$dx_2=13$$

As expected from Table 27, y_2 decreases and x_2 increases. So we can conclude that

TABLE 27. Result of sign analysis for Example 4.2.4

$[dx_1]$	$[dx_2]$	$[dy_1]$	$[dy_2]$
+	+	−	−
+	+	+	+
+	−	+	+
+	0	+	+
+	+	+	−
+	−	+	−
+	0	+	−
+	+	+	0
+	−	+	0
+	0	+	0
+	+	0	−

If we decrease y_1 *1 unit and decrease* y_2 *9 units, then we will observe a 1 unit increase in* x_1 *and a 13 units increase in* x_2.

What this example tells us that in some circumstances sign analysis and value analysis must go hand in hand for a better understanding of the system.

As a final note, the technique we have presented here is based on the use of gains. The method to compute gains (or marginal values) has been well-known and used widely in the economics literature for a long time (e.g. [1,4,13]). It is known as the method of *comparative statics*. Comparative statics presumes that the system is initially at equilibrium and examines how the equilibrium values of endogenous variables respond to a change in one or more of the parameters, that is in which direction they change and establish a new equilibrium to match the new configuration of the parameters.

4.3. Elasticities

In addition to gains, another value that can be of interest is called *elasticity*. Elasticities can be computed from gains. Elasticity of an endogenous variable x with respect to an exogenous variable y, E_{xy}, is defined as

$$E_{xy}=\frac{\frac{\partial x}{x}}{\frac{\partial y}{y}}=\frac{y}{x}\,\frac{\partial x}{\partial y} \tag{4.12}$$

and it represents the relative change $\frac{\partial x}{x}$ in the value of x for a relative change $\frac{\partial y}{y}$ in the value of y.

As an example, we can calculate elasticies for the example of Section 4.1 by using (4.8) and (4.9) as

$$E_{x_1y_1}=\frac{y_1}{x_1}\,\frac{\partial x_1}{\partial y_1}=\frac{8}{5}=1.6$$

$$E_{x_2y_1}=\frac{y_1}{x_2}\,\frac{\partial x_2}{\partial y_1}=\frac{5}{2}=2.5$$

$$E_{x_1y_2}=\frac{y_2}{x_1}\,\frac{\partial x_1}{\partial y_2}=-\frac{3}{5}=-0.6$$

$$E_{x_2y_2}=\frac{y_2}{x_2}\,\frac{\partial x_2}{\partial y_2}=-3$$

For example, elasticity of x_1 wrt y_1 means that although the change in x_1 is 8 times the change in y_1 (from (4.9)) for the particular system state given in (4.8), the corresponding relative change is only 1.6.

Elasticities are useful in cases where the parameters represent quantities that are very different in magnitude. For example, if the typical values that an endogenous variable, say x, can take on are in the order of 0.1, the typical values that an exogenous variable, say y, can take on are in the order of 100, and if a change in y causes a change in the same amount in x, the gain $\frac{\partial x}{\partial y}$ is 1 although the elasticity of x wrt y is almost 1000. In this case the value of the elasticity is more useful than the value of gain. This becomes clear if we consider that, for instance, a change of 0.1 in y causes a change of 0.1 in x, but such a change is almost negligible for y (compared to 100) although it doubles the value of x.

V. ILLUSTRATION OF THE TECHNIQUES

In this section, we will illustrate the techniques presented in the previous sections by using a complete system taken from [14]. Consider the system shown in Figure 9. This is a piping system of the form of a T junction with a pump at the left-hand side supplying a constant flow rate Q_F to two locations through two valves, discharging into pressures P_1 and P_2, respectively. The mathematical model of the system is given as follows:

Equations

$Q_F = Q_1 + Q_2$

$Q_1 = C_{v1}\sqrt{P_F - P_1}$

$Q_2 = C_{v2}\sqrt{P_F - P_2}$

Parameters

Q_F	constant flow rate of the pump (exo)
Q_1	flow rate of the first valve (endo)
Q_2	flow rate of the second valve (endo)
C_{v1}	valve constant for the first valve (const)
P_F	discharging pressure of the pump (endo)
P_1	pressure at the right-hand side of the first valve (exo)
C_{v2}	valve constant for the second valve (const)
P_2	pressure at the right-hand side of the second valve (exo)

The first equation above can be considered as the material balance for the T junction and the last two as the material balances for the two valves. We assume that P_F is

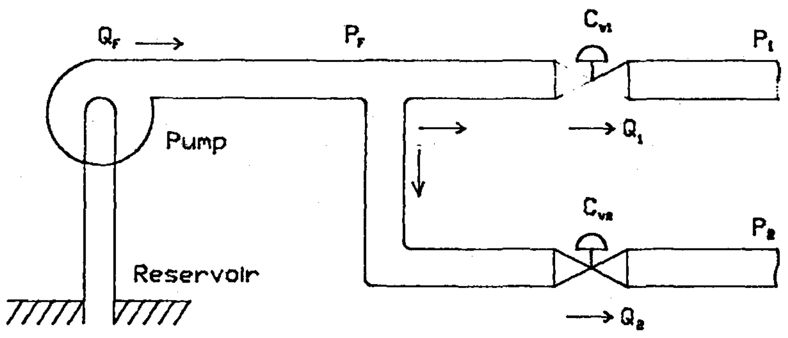

FIGURE 9. Example system used in Section 5

greater than P_1 and P_2 and the flows through the valves are in the directions as indicated on the Figure. (P_F-P_1) and (P_F-P_2) are called pressure drops across the valves 1 and 2, respectively.

The functions can be written in closed form as

$$\begin{aligned}
F_1(Q_F,Q_1,Q_2) &= Q_F-Q_1-Q_2 = 0\\
F_2(Q_1,C_{v1},P_F,P_1) &= Q_1-C_{v1}\sqrt{P_F-P_1} = 0\\
F_3(Q_2,C_{v2},P_F,P_2) &= Q_2-C_{v2}\sqrt{P_F-P_2} = 0
\end{aligned}$$

Note that the determination of the types of the parameters and the types of the variables depends on the choice of the model-builder. For this system, Q_F, P_1, and P_2 are treated as exogenous variables; Q_1, Q_2, and P_F as endogenous variables; and C_{v1} and C_{v2} as constants for physical considerations.

Now as the system is given, we can begin the analysis. However, before this process let us discuss the application of the assembling concept which was explained in Section 2 on this example.

5.1. Assembling

Suppose that a valve system, shown in Figure 3, is modeled as follows

Equations

$Q=C_v\sqrt{P_1-P_2}$

Parameters

Q	flow rate through the valve (endo)
C_v	valve constant (const)
P_1	pressure at the left-hand side of the valve (exo)
P_2	pressure at the right-hand side of the valve (exo)

We assume that P_1 is greater than P_2 and the direction of the flow is from P_1 to P_2.

Now suppose that a pump system made up of two branches, as shown in Figure 10, is modeled as follows

Equations

$Q_{in}=Q_{out1}+Q_{out2}$

Parameters

Q_{in}	flow rate of the pump (exo)
Q_{out1}	flow rate of the first branch (endo)
Q_{out2}	flow rate of the second branch (endo)

For the purpose of reference, let us name the valve model and the pump model as *Valve* and *Pump*, respectively.

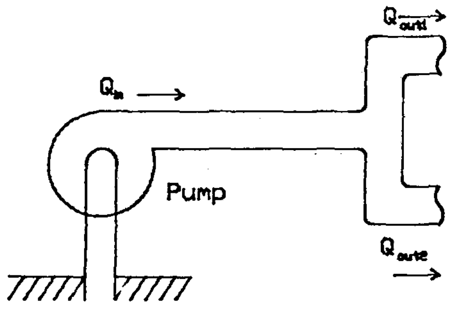

FIGURE 10. A pump system of two branches

Having these two subsystems, we can now construct the model of the system shown in Figure 9 as in the following informal notation:

Subsystems : *Pump*

$$Q_{in}=Q_{out1}+Q_{out2}$$

Valve (call it *Valve-1*)

$$Q=C_v\sqrt{P_1-P_2}$$

Valve (call it *Valve-2*)

$$Q=C_v\sqrt{P_1-P_2}$$

Parameter names & types

Pump	:	Q_{in}	➡	Q_F
		Q_{out1}	➡	Q_1
		Q_{out2}	➡	Q_2
Valve-1	:	Q	➡	Q_1
		C_v	➡	C_{v1}
		P_1	➡	P_F (endo)
		P_2	➡	P_1
Valve-2	:	Q	➡	Q_2
		C_v	➡	C_{v2}
		P_1	➡	P_F
		P_2	➡	P_2

We use the pump model and two copies of the valve model, referred to as *Valve-1* and *Valve-2*. The parameters are renamed as shown above. Also the types of those parameters whose types are different in the submodel and in the resulting model are changed, the other parameters use their original types. The result of this assembling process is the model given in the previous section.

Note that the identification of the parameters is implicitly done during renaming. For example, by giving the same name for Q_{out1} of *Pump* and Q of *Valve-1*, we indicate that the two refer to the same identity.

5.2. Sign Analysis of Total Differentials

To begin sign analysis, we first form the sign tables for each of the modeling equations. For this purpose we calculate the partial derivatives and express them in terms of the Information matrix of the system,

$$
I = \begin{array}{c|cccccccc}
 & Q_F & Q_1 & Q_2 & C_{v1} & C_{v2} & P_F & P_1 & P_2 \\
\hline
F_1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\
F_2 & 0 & 1 & 0 & -\sqrt{P_F-P_1} & 0 & \frac{C_{v1}}{2\sqrt{P_F-P_1}} & \frac{C_{v1}}{2\sqrt{P_F-P_1}} & 0 \\
F_3 & 0 & 0 & 1 & 0 & -\sqrt{P_F-P_2} & \frac{C_{v2}}{2\sqrt{P_F-P_2}} & 0 & \frac{C_{v2}}{2\sqrt{P_F-P_2}}
\end{array}
$$

which is an m*n matrix of elements $\frac{\partial F_i}{\partial z_j}$ with i=1,...,m and j=1,...,n, where m (number of functions) is 3 and n (number of parameters) is 8 for this example.

We assume, as indicated above, that (P_F-P_1) and (P_F-P_2) denote positive quantities because otherwise the square roots would have no real values. Based on this assumption, the I matrix is converted to the I_S matrix

$$I_S = \begin{array}{c} \\ F_1 \\ F_2 \\ F_3 \end{array} \begin{array}{cccccccc} Q_F & Q_1 & Q_2 & C_{v1} & C_{v2} & P_F & P_1 & P_2 \\ \end{array}$$

$$\begin{array}{c|cccccccc} & Q_F & Q_1 & Q_2 & C_{v1} & C_{v2} & P_F & P_1 & P_2 \\ \hline F_1 & + & - & - & 0 & 0 & 0 & 0 & 0 \\ F_2 & 0 & + & 0 & - & 0 & [\overline{C_{v1}}] & [C_{v1}] & 0 \\ F_3 & 0 & 0 & + & 0 & - & [\overline{C_{v2}}] & 0 & [C_{v2}] \end{array}$$

based on the fact that $[\sqrt{P_F-P_1}\,]=+$ and $[\sqrt{P_F-P_2}\,]=+$.

The first equation has 3 parameters and the other two equations have 4 parameters each. Making use of Tables 3 and 4 for a three- and a four-parameter function, respectively, we can construct the sign tables of these functions as in Tables 28, 29, and 30.

TABLE 28. Sign table of $F=Q_F-Q_1-Q_2=0$

$[dQ_F]$	$[dQ_1]$	$[dQ_2]$
+	−	+
+	+	−
+	+	+
+	+	0
+	0	+
−	−	−
−	−	+
−	−	0
−	+	−
−	0	−
0	−	+
0	+	−
0	0	0

Tables 29 and 30 contain entries expressed in terms of C_{v1} and C_{v2}. In order to be able to convert these into pure sign tables, we must know the signs of C_{v1} and C_{v2}. Suppose that C_{v1} and C_{v2} are both positive[7], thus $[C_{v1}]=+$ and $[C_{v2}]=+$. Substituting these into Tables 29 and 30 results in Tables 31 and 32.

Now we can perform sign analysis by using Tables 28, 31, and 32. Call these tables T_1, T_2, and T_3, respectively, for functions F_1, F_2, and F_3. Let us first generate a notation for ease of presentation. We will use, for a parameter z, the following terms

$\{z\}=+$	:	the value of z is increased
$\{z\}=-$	:	the value of z is decreased
$\{z\}=0$	:	the value of z remains unchanged
$\{z\}=?$	:	the direction of change of z is unknown.

The first three represent terms we have already been using and the last one represents a parameter whose direction of change is to be determined by the analysis. A perturbation formula, θ, is formed by taking the conjunction of the parameters whose signs are known.

Below we give some examples together with their interpretations. Keep in mind that all the following results are valid for the case C_{v1} and C_{v2} are positive.

[7] In fact, in the system being examined, these are the only possibilities that make sense since C_{v1} and C_{v2} represent valve constants which can not be negative physically. Therefore this assumption does not bring any restriction into the system space.

TABLE 29. Sign table of $F=Q_1-C_{v1}\sqrt{P_F-P_1}=0$

$[dQ_1]$	$[dC_{v1}]$	$[dP_F]$	$[dP_1]$
+	–	$[\overline{C}_{v1}]$	$[\overline{C}_{v1}]$
+	–	$[C_{v1}]$	$[C_{v1}]$
+	–	$[C_{v1}]$	$[\overline{C}_{v1}]$
+	–	$[C_{v1}]$	0
+	–	0	$[\overline{C}_{v1}]$
+	+	$[\overline{C}_{v1}]$	$[C_{v1}]$
+	+	$[\overline{C}_{v1}]$	$[\overline{C}_{v1}]$
+	+	$[\overline{C}_{v1}]$	0
+	+	$[C_{v1}]$	$[C_{v1}]$
+	+	$[C_{v1}]$	$[\overline{C}_{v1}]$
+	+	$[C_{v1}]$	0
+	+	0	$[C_{v1}]$
+	+	0	$[\overline{C}_{v1}]$
+	+	0	0
+	0	$[\overline{C}_{v1}]$	$[\overline{C}_{v1}]$
+	0	$[C_{v1}]$	$[C_{v1}]$
+	0	$[C_{v1}]$	$[\overline{C}_{v1}]$
+	0	$[C_{v1}]$	0
+	0	0	$[\overline{C}_{v1}]$
–	–	$[\overline{C}_{v1}]$	$[C_{v1}]$
–	–	$[\overline{C}_{v1}]$	$[\overline{C}_{v1}]$
–	–	$[\overline{C}_{v1}]$	0
–	–	$[C_{v1}]$	$[C_{v1}]$
–	–	$[C_{v1}]$	$[\overline{C}_{v1}]$
–	–	$[C_{v1}]$	0
–	–	0	$[C_{v1}]$
–	–	0	$[\overline{C}_{v1}]$
–	–	0	0
–	+	$[\overline{C}_{v1}]$	$[C_{v1}]$
–	+	$[\overline{C}_{v1}]$	$[\overline{C}_{v1}]$
–	+	$[\overline{C}_{v1}]$	0
–	+	$[C_{v1}]$	$[C_{v1}]$
–	+	0	$[C_{v1}]$
–	0	$[\overline{C}_{v1}]$	$[C_{v1}]$
–	0	$[\overline{C}_{v1}]$	$[\overline{C}_{v1}]$
–	0	$[\overline{C}_{v1}]$	0
–	0	$[C_{v1}]$	$[C_{v1}]$
–	0	0	$[C_{v1}]$
0	–	$[\overline{C}_{v1}]$	$[\overline{C}_{v1}]$
0	–	$[C_{v1}]$	$[C_{v1}]$
0	–	$[C_{v1}]$	$[\overline{C}_{v1}]$
0	–	$[C_{v1}]$	0
0	–	0	$[\overline{C}_{v1}]$
0	+	$[\overline{C}_{v1}]$	$[C_{v1}]$
0	+	$[\overline{C}_{v1}]$	$[\overline{C}_{v1}]$
0	+	$[\overline{C}_{v1}]$	0
0	+	$[C_{v1}]$	$[C_{v1}]$
0	+	0	$[C_{v1}]$
0	0	$[\overline{C}_{v1}]$	$[\overline{C}_{v1}]$
0	0	$[C_{v1}]$	$[C_{v1}]$
0	0	0	0

TABLE 30. Sign table of $F=Q_2-C_{v2}\sqrt{P_F-P_2}=0$

$[dQ_2]$	$[dC_{v2}]$	$[dP_F]$	$[dP_2]$
+	−	$[\overline{C_{v2}}]$	$[\overline{C_{v2}}]$
+	−	$[C_{v2}]$	$[C_{v2}]$
+	−	$[C_{v2}]$	$[\overline{C_{v2}}]$
+	−	$[C_{v2}]$	0
+	−	0	$[\overline{C_{v2}}]$
+	+	$[\overline{C_{v2}}]$	$[C_{v2}]$
+	+	$[\overline{C_{v2}}]$	$[\overline{C_{v2}}]$
+	+	$[\overline{C_{v2}}]$	0
+	+	$[C_{v2}]$	$[C_{v2}]$
+	+	$[C_{v2}]$	$[\overline{C_{v2}}]$
+	+	$[C_{v2}]$	0
+	+	0	$[C_{v2}]$
+	+	0	$[\overline{C_{v2}}]$
+	+	0	0
+	0	$[\overline{C_{v2}}]$	$[\overline{C_{v2}}]$
+	0	$[C_{v2}]$	$[C_{v2}]$
+	0	$[C_{v2}]$	$[\overline{C_{v2}}]$
+	0	$[C_{v2}]$	0
+	0	0	$[\overline{C_{v2}}]$
−	−	$[\overline{C_{v2}}]$	$[C_{v2}]$
−	−	$[\overline{C_{v2}}]$	$[\overline{C_{v2}}]$
−	−	$[\overline{C_{v2}}]$	0
−	−	$[C_{v2}]$	$[C_{v2}]$
−	−	$[C_{v2}]$	$[\overline{C_{v2}}]$
−	−	$[C_{v2}]$	0
−	−	0	$[C_{v2}]$
−	−	0	$[\overline{C_{v2}}]$
−	−	0	0
−	+	$[\overline{C_{v2}}]$	$[C_{v2}]$
−	+	$[\overline{C_{v2}}]$	$[\overline{C_{v2}}]$
−	+	$[\overline{C_{v2}}]$	0
−	+	$[C_{v2}]$	$[C_{v2}]$
−	+	0	$[C_{v2}]$
−	0	$[\overline{C_{v2}}]$	$[C_{v2}]$
−	0	$[\overline{C_{v2}}]$	$[\overline{C_{v2}}]$
−	0	$[\overline{C_{v2}}]$	0
−	0	$[C_{v2}]$	$[C_{v2}]$
−	0	0	$[C_{v2}]$
0	−	$[\overline{C_{v2}}]$	$[\overline{C_{v2}}]$
0	−	$[C_{v2}]$	$[C_{v2}]$
0	−	$[C_{v2}]$	$[\overline{C_{v2}}]$
0	−	$[C_{v2}]$	0
0	−	0	$[\overline{C_{v2}}]$
0	+	$[\overline{C_{v2}}]$	$[C_{v2}]$
0	+	$[\overline{C_{v2}}]$	$[\overline{C_{v2}}]$
0	+	$[\overline{C_{v2}}]$	0
0	+	$[C_{v2}]$	$[C_{v2}]$
0	+	0	$[C_{v2}]$
0	0	$[\overline{C_{v2}}]$	$[\overline{C_{v2}}]$
0	0	$[C_{v2}]$	$[C_{v2}]$
0	0	0	0

TABLE 31. Pure sign table of $F=Q_1-C_{v1}\sqrt{P_F-P_1}=0$

$[dQ_1]$	$[dC_{v1}]$	$[dP_F]$	$[dP_1]$	$[dQ_1]$	$[dC_{v1}]$	$[dP_F]$	$[dP_1]$
+	−	−	−	−	−	0	−
+	−	+	+	−	−	0	0
+	−	+	−	−	+	−	+
+	−	+	0	−	+	−	−
+	−	0	−	−	+	−	0
+	+	−	+	−	+	+	+
+	+	−	−	−	+	0	+
+	+	−	0	−	0	−	+
+	+	+	+	−	0	−	−
+	+	+	−	−	0	−	0
+	+	+	0	−	0	+	+
+	+	0	+	−	0	0	+
+	+	0	−	0	−	−	−
+	+	0	0	0	−	+	+
+	0	−	−	0	−	+	−
+	0	+	+	0	−	+	0
+	0	+	−	0	−	0	−
+	0	+	0	0	+	−	+
+	0	0	−	0	+	−	−
−	−	−	+	0	+	−	0
−	−	−	−	0	+	+	+
−	−	−	0	0	+	0	+
−	−	+	+	0	0	−	−
−	−	+	−	0	0	+	+
−	−	+	0	0	0	0	0
−	−	0	+				

TABLE 32. Pure sign table of $F=Q_2-C_{v2}\sqrt{P_F-P_2}=0$

$[dQ_2]$	$[dC_{v2}]$	$[dP_F]$	$[dP_2]$
+	−	−	−
+	−	+	+
+	−	+	−
+	−	+	0
+	−	0	−
+	+	−	+
+	+	−	−
+	+	−	0
+	+	+	+
+	+	+	−
+	+	+	0
+	+	0	+
+	+	0	−
+	+	0	0
+	0	−	−
+	0	+	+
+	0	+	−
+	0	+	0
+	0	0	−
−	−	−	+
−	−	−	−
−	−	−	0
−	−	+	+
−	−	+	−
−	−	+	0
−	−	0	+
−	−	0	−
−	−	0	0
−	+	−	+
−	+	−	−
−	+	−	0
−	+	+	+
−	+	0	+
−	0	−	+
−	0	−	−
−	0	−	0
−	0	+	+
−	0	0	+
0	−	−	−
0	−	+	+
0	−	+	−
0	−	+	0
0	−	0	−
0	+	−	+
0	+	−	−
0	+	−	0
0	+	+	+
0	+	0	+
0	0	−	−
0	0	+	+
0	0	0	0

Example 5.2.1. We increase Q_F while keeping other exogenous variables and constants unchanged. How do the endogenous variables change?

$\{Q_F\}=+$ $\{C_{v1}\}=\{C_{v2}\}=\{P_1\}=\{P_2\}=0$ $\{Q_1\}=\{Q_2\}=\{P_F\}=?$

$\theta=(\{Q_F\}=+\wedge\{C_{v1}\}=0\wedge\{C_{v2}\}=0\wedge\{P_1\}=0\wedge\{P_2\}=0)$

$T_1 \underset{\theta}{\bowtie} T_2 \underset{\theta}{\bowtie} T_3$ is shown in Table 33.

So, there is one solution: $\{Q_1\}=\{Q_2\}=\{P_F\}=+$.

Let us give an explanation of how the result is found. T_1 has 5 rows satisfying $\{Q_F\}=+$, T_2 has 3 rows satisfying $\{C_{v1}\}=0\wedge\{P_1\}=0$, and T_3 has 3 rows satisfying $\{C_{v2}\}=0\wedge\{P_2\}=0$, as shown in Figure 11 and the tables formed by these rows are named as T_1', T_2', and T_3', respectively. $[dQ_1]$ and $[dQ_2]$ must be the same in order for $[dP_F]$ of T_2' and $[dP_F]$ of T_3' do not conflict. That is, $[dQ_1]=[dQ_2]=+$, $[dQ_1]=[dQ_2]=-$, or $[dQ_1]=[dQ_2]=0$. Only the first one of these exists in T_1'. Thus Q_1 and Q_2 both increase, and it is found from the corresponding row of either T_2' or T_3' that P_F must also increase.

TABLE 33. Result of Example 5.2.1

$[dQ_F]$	$[dQ_1]$	$[dQ_2]$	$[dC_{v1}]$	$[dP_F]$	$[dP_1]$	$[dC_{v2}]$	$[dP_2]$
+	+	+	0	+	0	0	0

$[dQ_F]$	$[dQ_1]$	$[dQ_2]$
+	−	+
+	+	−
+	+	+
+	+	0
+	0	+

$[dQ_1]$	$[dC_{v1}]$	$[dP_F]$	$[dP_1]$
+	0	+	0
−	0	−	0
0	0	0	0

$[dQ_2]$	$[dC_{v2}]$	$[dP_F]$	$[dP_2]$
+	0	+	0
−	0	−	0
0	0	0	0

FIGURE 11. Reduced tables for the increase in Q_F

Example 5.2.2. We decrease C_{v1} while keeping other exogenous variables unchanged. How do the endogenous variables change?

$\{C_{v1}\}=-\quad \{Q_F\}=\{C_{v2}\}=\{P_1\}=\{P_2\}=0\quad \{Q_1\}=\{Q_2\}=\{P_F\}=?$

$\theta=(\{Q_F\}=0\wedge\{C_{v1}\}=-\wedge\{C_{v2}\}=0\wedge\{P_1\}=0\wedge\{P_2\}=0)$

$T_1 \underset{\theta}{\rhd\!\!\lhd} T_2 \underset{\theta}{\rhd\!\!\lhd} T_3$ is shown in Table 34.

Q_1 decreases while Q_2 and P_F increase.

Example 5.2.3. We increase Q_F and decrease C_{v1} while keeping other exogenous variables unchanged. How do the endogenous variables change?

$\{Q_F\}=+\quad \{C_{v1}\}-\quad \{C_{v2}\}=\{P_1\}=\{P_2\}=0\quad \{Q_1\}=\{Q_2\}=\{P_F\}=?$

$\theta=(\{Q_F\}=+\wedge\{C_{v1}\}=-\wedge\{C_{v2}\}=0\wedge\{P_1\}=0\wedge\{P_2\}=0)$

$T_1 \underset{\theta}{\rhd\!\!\lhd} T_2 \underset{\theta}{\rhd\!\!\lhd} T_3$ is shown in Table 35.

Q_2 and P_F increase while Q_1 may change in any direction.

The existence of three solutions in this example means that with different system states and with different amounts of perturbation (i.e. dQ_F and dC_{v1}), it is possible for Q_1 to change in any one of the directions.

TABLE 34. Result of Example 5.2.2

$[dQ_F]$	$[dQ_1]$	$[dQ_2]$	$[dC_{v1}]$	$[dP_F]$	$[dP_1]$	$[dC_{v2}]$	$[dP_2]$
0	−	+	−	+	0	0	0

TABLE 35. Result of Example 5.2.3

$[dQ_F]$	$[dQ_1]$	$[dQ_2]$	$[dC_{v1}]$	$[dP_F]$	$[dP_1]$	$[dC_{v2}]$	$[dP_2]$
+	+	+	−	+	0	0	0
+	−	+	−	+	0	0	0
+	0	+	−	+	0	0	0

Example 5.2.4. Suppose that we are able to increase the endogenous variable Q_1 without affecting the exogenous variables. How do Q_2 and P_F change?

$\{Q_1\}=+ \quad \{Q_F\}=\{C_{v1}\}=\{C_{v2}\}=\{P_1\}=\{P_2\}=0 \quad \{Q_2\}=\{P_F\}=?$

$\theta=(\{Q_F\}=0\wedge\{C_{v1}\}=0\wedge\{C_{v2}\}=0\wedge\{P_1\}=0\wedge\{P_2\}=0\wedge\{Q_1\}=+)$

$T_1 \bowtie_\theta T_2 \bowtie_\theta T_3$ results in an empty table.

This means that, in a particular system state (i.e. without changing the values of exogenous variables and constants), even if we are able to change the value of Q_1, it will return immediately to its original value. In other words, given the values of the exogenous variables and constants, there is a unique solution for the values of the endogenous variables and we cannot alter those values without changing the values of the exogenous variables and constants.

It is worth noting here that if the value of C_{v1} were declared as negative at the beginning of the analysis (in which case $[C_{v1}]=-$ so T_2, which was produced from Table 29, would be different) then the above join operation would result in a single solution: Q_2 and P_F decrease. This means that without changing the values of the exogenous variables and constants, endogenous variables may take different values. In other words, the system is not stable.

Example 5.2.5. We increase Q_F while keeping other exogenous variables unchanged and we want Q_1 to increase. Is this possible, and if so how do Q_2 and P_F change?

$\{Q_F\}=\{Q_1\}=+ \quad \{C_{v1}\}=\{C_{v2}\}=\{P_1\}=\{P_2\}=0 \quad \{Q_2\}=\{P_F\}=?$

$\theta=(\{Q_F\}=+\wedge\{C_{v1}\}=0\wedge\{C_{v2}\}=0\wedge\{P_1\}=0\wedge\{P_2\}=0\wedge\{Q_1\}=+)$

$T_1 \bowtie_\theta T_2 \bowtie_\theta T_3$ results in Table 33.

We conclude that this is possible and Q_2 and P_F increase as well.

This question is similar to the first one. In the first question we did not have a restriction on Q_1 but the result has shown that Q_1 being increased is the only solution. Thus these two questions give the same signs for Q_2 and P_F.

Example 5.2.6. . We want P_F to increase and Q_1 and Q_2 remain unchanged. How must exogenous variables and constants be changed in order to satisfy this condition? (or, It is observed that P_F is increasing while Q_1 and Q_2 remain unchanged. What can cause this?)

$\{P_F\}=+ \quad \{Q_1\}=\{Q_2\}=0 \quad \{Q_F\}=\{C_{v1}\}=\{C_{v2}\}=\{P_1\}=\{P_2\}=?$

$\ominus=(\{Q_1\}=0 \wedge \{Q_2\}=0 \wedge \{P_F\}=+)$

$T_1 \underset{\ominus}{\rhd\!\!-\!\!\lhd} T_2 \underset{\ominus}{\rhd\!\!-\!\!\lhd} T_3$ is shown in Table 36.

There are 25 possibilities. We can summarize the result as follows:

TABLE 36. Result of Example 5.2.6

$[dQ_F]$	$[dQ_1]$	$[dQ_2]$	$[dC_{v1}]$	$[dP_F]$	$[dP_1]$	$[dC_{v2}]$	$[dP_2]$
0	0	0	−	+	+	−	+
0	0	0	−	+	+	−	−
0	0	0	−	+	+	−	0
0	0	0	−	+	+	+	+
0	0	0	−	+	+	0	+
0	0	0	−	+	−	−	+
0	0	0	−	+	−	−	−
0	0	0	−	+	−	−	0
0	0	0	−	+	−	+	+
0	0	0	−	+	−	0	+
0	0	0	−	+	0	−	+
0	0	0	−	+	0	−	−
0	0	0	−	+	0	−	0
0	0	0	−	+	0	+	+
0	0	0	−	+	0	0	+
0	0	0	+	+	+	−	+
0	0	0	+	+	+	−	−
0	0	0	+	+	+	−	0
0	0	0	+	+	+	+	+
0	0	0	+	+	+	0	+
0	0	0	0	+	+	−	+
0	0	0	0	+	+	−	−
0	0	0	0	+	+	−	0
0	0	0	0	+	+	+	+
0	0	0	0	+	+	0	+

Q_F must not change. If C_{v1} is decreased then P_1 can be changed in any direction, otherwise P_1 must be increased. Similarly, if C_{v2} is decreased then P_2 can be changed in any direction, otherwise P_2 must be increased.

Example 5.2.7. We increase P_1 while keeping Q_F, C_{v1}, and C_{v2} unchanged. We want P_F not to change. How must we change P_2 to satisfy this condition and how do Q_1 and Q_2 change?

$\{P_1\}=+ \quad \{Q_F\}=\{C_{v1}\}=\{C_{v2}\}=\{P_F\}=0 \quad \{P_2\}=\{Q_1\}=\{Q_2\}=?$

$\theta=(\{Q_F\}=0\wedge\{C_{v1}\}=0\wedge\{C_{v2}\}=0\wedge\{P_1\}=+\wedge\{P_F\}=0)$

$T_1 \underset{\theta}{\sqsubset\!\rhd\!\lhd} T_2 \underset{\theta}{\sqsubset\!\rhd\!\lhd} T_3$ is shown in Table 37.

We must decrease P_2. Q_1 will decrease and Q_2 will increase.

Example 5.2.8. We decrease Q_F and increase C_{v1} while keeping C_{v2} unchanged. We want Q_1 to decrease and P_F not to change. How must P_1 and P_2 be changed to satisfy this condition and how does Q_2 change?

$\{Q_F\}=\{Q_1\}=- \quad \{C_{v1}\}=+ \quad \{C_{v2}\}=\{P_F\}=0 \quad \{P_1\}=\{P_2\}=\{Q_2\}=?$

$\theta=(\{Q_F\}=-\wedge\{C_{v1}\}=+\wedge\{C_{v2}\}=0\wedge\{Q_1\}=-\wedge\{P_F\}=0)$

$T_1 \underset{\theta}{\sqsubset\!\rhd\!\lhd} T_2 \underset{\theta}{\sqsubset\!\rhd\!\lhd} T_3$ is shown in Table 38.

We must increase P_1 and we can change P_2 in any direction. Q_2 will change in the opposite direction of P_2.

TABLE 37. Result of Example 5.2.7

$[dQ_F]$	$[dQ_1]$	$[dQ_2]$	$[dC_{v1}]$	$[dP_F]$	$[dP_1]$	$[dC_{v2}]$	$[dP_2]$
0	–	+	0	0	+	0	–

TABLE 38. Result of Example 5.2.8

$[dQ_F]$	$[dQ_1]$	$[dQ_2]$	$[dC_{v1}]$	$[dP_F]$	$[dP_1]$	$[dC_{v2}]$	$[dP_2]$
–	–	–	+	0	+	0	+
–	–	+	+	0	+	0	–
–	–	0	+	0	+	0	0

We can perform sign analysis in a different way also. Suppose that Q_F is increased, C_{v1} and C_{v2} are kept unchanged, and Q_1 is decreased. Then Tables 28, 31, and 32 reduce to the sign tables shown in Tables 39, 40, and 41, respectively. (We assume, as above, that $[C_{v1}]=+$ and $[C_{v2}]=+$, so use Tables 28, 31, and 32). Wee see from Table 39 that there is one possibility for Q_2, namely, it increases. So, Table 41 can be further simplified, yielding Table 42.

Using Tables 39, 40, and 42 (call T_1, T_2, and T_3, respectively), we can now extract the following rules, for the situation where $[C_{v1}]=+$ and $[C_{v2}]=+$:

If Q_F is increased and Q_1 is decreased then Q_2 will increase (from T_1).

If Q_1 is decreased while keeping C_{v1} unchanged then if P_F increases then P_1 should increase (from T_2).

If Q_1 is decreased while keeping C_{v1} unchanged then if P_F decreases then P_1 may increase, decrease, or stay unchanged (from T_2).

If Q_1 is decreased while keeping C_{v1} unchanged then if P_F increases or stays unchanged then P_1 should increase (from T_2).

If Q_2 is increased while keeping C_{v2} unchanged then if $P_{F@}$ decreases then P_2 should decrease (from T_3).

If Q_2 is increased while keeping C_{v1} unchanged then if P_F increases then P_2 may increase, decrease, or stay unchanged (from T_3).

If Q_2 is increased while keeping C_{v1} unchanged then if P_F decreases or stays unchanged then P_2 should decrease (from T_3).

TABLE 39. Reduced form of T_1

$[dQ_F]$	$[dQ_1]$	$[dQ_2]$
+	−	+

TABLE 40. Reduced form of T_2

$[dQ_1]$	$[dC_{v1}]$	$[dP_F]$	$[dP_1]$
−	0	−	+
−	0	−	−
−	0	−	0
−	0	+	+
−	0	0	+

TABLE 41. Reduced form of T_3

$[dQ_2]$	$[dC_{v2}]$	$[dP_F]$	$[dP_2]$
+	0	−	−
+	0	+	+
+	0	+	−
+	0	+	0
+	0	0	−
−	0	−	+
−	0	−	−
−	0	−	0
−	0	+	+
−	0	0	+
0	0	−	−
0	0	+	+
0	0	0	0

TABLE 42. Reduced form of Table 41

$[dQ_2]$	$[dC_{v2}]$	$[dP_F]$	$[dP_2]$
+	0	−	−
+	0	+	+
+	0	+	−
+	0	+	0
+	0	0	−

If Q_F is increased and Q_1 is decreased then Q_2 will increase and if C_{v2} is kept unchanged then if P_F decreases then P_2 should decrease (from T_1 and T_3).

This analysis is exactly identical to the one presented earlier. The two both yield the same results. The difference is that, in the latter one, the join operation is done implicitly by reducing the tables first and then searching for the possibilities. In fact, once the sign tables are formed, we can perform sign analysis on these tables in any way we desire.

5.3. Sign Analysis of Gains

As explained in Section 3.3, to begin sign analysis of gains, we write the equations in the form of (3.10), for an exogenous variable. Suppose that we perform the analysis for Q_F,

$$[\frac{\partial F_1}{\partial Q_F}\frac{\partial Q_F}{\partial Q_F}] + [\frac{\partial F_1}{\partial Q_1}\frac{\partial Q_1}{\partial Q_F}] + [\frac{\partial F_1}{\partial Q_2}\frac{\partial Q_2}{\partial Q_F}] + [\frac{\partial F_1}{\partial C_{v1}}\frac{\partial C_{v1}}{\partial Q_F}] + [\frac{\partial F_1}{\partial C_{v2}}\frac{\partial C_{v2}}{\partial Q_F}] + [\frac{\partial F_1}{\partial P_F}\frac{\partial P_F}{\partial Q_F}] + [\frac{\partial F_1}{\partial P_1}\frac{\partial P_1}{\partial Q_F}] + [\frac{\partial F_1}{\partial P_2}\frac{\partial P_2}{\partial Q_F}] = 0$$

$$[\frac{\partial F_2}{\partial Q_F}\frac{\partial Q_F}{\partial Q_F}] + [\frac{\partial F_2}{\partial Q_1}\frac{\partial Q_1}{\partial Q_F}] + [\frac{\partial F_2}{\partial Q_2}\frac{\partial Q_2}{\partial Q_F}] + [\frac{\partial F_2}{\partial C_{v1}}\frac{\partial C_{v1}}{\partial Q_F}] + [\frac{\partial F_2}{\partial C_{v2}}\frac{\partial C_{v2}}{\partial Q_F}] + [\frac{\partial F_2}{\partial P_F}\frac{\partial P_F}{\partial Q_F}] + [\frac{\partial F_2}{\partial P_1}\frac{\partial P_1}{\partial Q_F}] + [\frac{\partial F_2}{\partial P_2}\frac{\partial P_2}{\partial Q_F}] = 0$$

$$[\frac{\partial F_3}{\partial Q_F}\frac{\partial Q_F}{\partial Q_F}] + [\frac{\partial F_3}{\partial Q_1}\frac{\partial Q_1}{\partial Q_F}] + [\frac{\partial F_3}{\partial Q_2}\frac{\partial Q_2}{\partial Q_F}] + [\frac{\partial F_3}{\partial C_{v1}}\frac{\partial C_{v1}}{\partial Q_F}] + [\frac{\partial F_3}{\partial C_{v2}}\frac{\partial C_{v2}}{\partial Q_F}] + [\frac{\partial F_3}{\partial P_F}\frac{\partial P_F}{\partial Q_F}] + [\frac{\partial F_3}{\partial P_1}\frac{\partial P_1}{\partial Q_F}] + [\frac{\partial F_3}{\partial P_2}\frac{\partial P_2}{\partial Q_F}] = 0$$

Using the I matrix, these simplify to

$$[\frac{\partial Q_F}{\partial Q_F}] - [\frac{\partial Q_1}{\partial Q_F}] - [\frac{\partial Q_2}{\partial Q_F}] = 0$$

$$[\frac{\partial Q_1}{\partial Q_F}] - [\sqrt{P_F - P_1} \frac{\partial C_{v1}}{\partial Q_F}] - [\frac{C_{v1}}{2\sqrt{P_F - P_1}} \frac{\partial P_F}{\partial Q_F}] + [\frac{C_{v1}}{2\sqrt{P_F - P_1}} \frac{\partial P_1}{\partial Q_F}] = 0$$

$$[\frac{\partial Q_2}{\partial Q_F}] - [\sqrt{P_F - P_2} \frac{\partial C_{v2}}{\partial Q_F}] - [\frac{C_{v2}}{2\sqrt{P_F - P_2}} \frac{\partial P_F}{\partial Q_F}] + [\frac{C_{v2}}{\sqrt{P_F - P_2}} \frac{\partial P_2}{\partial Q_F}] = 0$$

Now we form the sign tables for gains of the above equations. Assuming

$$[\sqrt{P_F - P_1}] = + \quad [\sqrt{P_F - P_2}] = + \quad [C_{v1}] = + \quad [C_{v2}] = +$$

as in the previous section, we get the pure sign tables shown in Tables 28, 31, and 32, except the column titles; the title [dz] must be replaced by $[\frac{\partial z}{\partial Q_F}]$ for $z \in \{Q_F, Q_1, Q_2, C_{v1}, P_F, P_1, C_{v2}, P_2\}$. As explained in Section 3.3, the equations for total differentials and gains possess the same coefficients, thus the sign tables are identical.

We require $[\frac{\partial Q_F}{\partial Q_F}] = +$, so Table 28 reduces to Table 43. Call Tables 43, 31, and 32, T_1, T_2, and T_3, respectively. Now let us answer the following questions. Keep in mind that C_{v1} and C_{v2} are positive.

TABLE 43. Reduced sign table of gains for $F = Q_F - Q_1 - Q_2 = 0$

$[\frac{\partial Q_F}{\partial Q_F}]$	$[\frac{\partial Q_1}{\partial Q_F}]$	$[\frac{\partial Q_2}{\partial Q_F}]$
+	−	+
+	+	−
+	+	+
+	+	0
+	0	+

Example 5.3.1. Q_F is changed. Other exogenous variables are kept unchanged. How do the endogenous variables change?

$$[\frac{\partial C_{v1}}{\partial Q_F}]=[\frac{\partial C_{v2}}{\partial Q_F}]=[\frac{\partial P_1}{\partial Q_F}]=[\frac{\partial P_2}{\partial Q_F}]=0 \qquad [\frac{\partial Q_1}{\partial Q_F}]=[\frac{\partial Q_2}{\partial Q_F}]=[\frac{\partial P_F}{\partial Q_F}]=?$$

$$\theta=([\frac{\partial C_{v1}}{\partial Q_F}]=0\wedge[\frac{\partial C_{v2}}{\partial Q_F}]=0\wedge[\frac{\partial P_1}{\partial Q_F}]=0\wedge[\frac{\partial P_2}{\partial Q_F}]=0)$$

$T_1\underset{\theta}{\rhd\!\lhd}T_2\underset{\theta}{\rhd\!\lhd}T_3$ is shown in Table 44.

Q_1, Q_2, and P_F change in the same direction as Q_F.

Example 5.3.2. Q_F is changed. C_{v1} is changed in the opposite direction of Q_F. Other exogenous variables are kept unchanged. How do the endogenous variables change?

$$[\frac{\partial C_{v1}}{\partial Q_F}]=- \qquad [\frac{\partial C_{v2}}{\partial Q_F}]=[\frac{\partial P_1}{\partial Q_F}]=[\frac{\partial P_2}{\partial Q_F}]=0 \qquad [\frac{\partial Q_1}{\partial Q_F}]=[\frac{\partial Q_2}{\partial Q_F}]=[\frac{\partial P_F}{\partial Q_F}]=?$$

$$\theta=([\frac{\partial C_{v1}}{\partial Q_F}]=-\wedge[\frac{\partial C_{v2}}{\partial Q_F}]=0\wedge[\frac{\partial P_1}{\partial Q_F}]=0\wedge[\frac{\partial P_2}{\partial Q_F}]=0)$$

$T_1\underset{\theta}{\rhd\!\lhd}T_2\underset{\theta}{\rhd\!\lhd}T_3$ is shown in Table 45.

Q_2 and P_F change in the same direction as Q_F, Q_1 may change in any direction.

Example 5.3.3. Q_F is changed. C_{v1} is changed in the opposite direction of Q_F. C_{v2} is kept unchanged. We want Q_1 to change in the same direction as Q_F and P_F not to change. How must P_1 and P_2 be changed to satisfy this condition and how does Q_2 changes?

$$[\frac{\partial Q_1}{\partial Q_F}]=+ \qquad [\frac{\partial C_{v1}}{\partial Q_F}]=- \qquad [\frac{\partial C_{v2}}{\partial Q_F}]=[\frac{\partial P_F}{\partial Q_F}]=0 \qquad [\frac{\partial P_1}{\partial Q_F}]=[\frac{\partial P_2}{\partial Q_F}]=[\frac{\partial Q_2}{\partial Q_F}]=?$$

$$\theta=([\frac{\partial Q_1}{\partial Q_F}]=+\wedge[\frac{\partial C_{v1}}{\partial Q_F}]=-\wedge[\frac{\partial C_{v2}}{\partial Q_F}]=0\wedge[\frac{\partial P_F}{\partial Q_F}]=0)$$

$T_1\underset{\theta}{\rhd\!\lhd}T_2\underset{\theta}{\rhd\!\lhd}T_3$ is shown in Table 46.

We must change P_1 in the opposite direction of Q_F, and we can change P_2 in any direction. Q_2 will change in the opposite direction of P_2.

TABLE 44. Result of Example 5.3.1

$[\frac{\partial Q_F}{\partial Q_F}]$	$[\frac{\partial Q_1}{\partial Q_F}]$	$[\frac{\partial Q_2}{\partial Q_F}]$	$[\frac{\partial C_{v1}}{\partial Q_F}]$	$[\frac{\partial P_F}{\partial Q_F}]$	$[\frac{\partial P_1}{\partial Q_F}]$	$[\frac{\partial C_{v2}}{\partial Q_F}]$	$[\frac{\partial P_2}{\partial Q_F}]$
+	+	+	0	+	0	0	0

TABLE 45. Result of Example 5.3.2

$[\frac{\partial Q_F}{\partial Q_F}]$	$[\frac{\partial Q_1}{\partial Q_F}]$	$[\frac{\partial Q_2}{\partial Q_F}]$	$[\frac{\partial C_{v1}}{\partial Q_F}]$	$[\frac{\partial P_F}{\partial Q_F}]$	$[\frac{\partial P_1}{\partial Q_F}]$	$[\frac{\partial C_{v2}}{\partial Q_F}]$	$[\frac{\partial P_2}{\partial Q_F}]$
+	+	+	−	+	0	0	0
+	−	+	−	+	0	0	0
+	0	+	−	+	0	0	0

TABLE 46. Result of Example 5.3.3

$[\frac{\partial Q_F}{\partial Q_F}]$	$[\frac{\partial Q_1}{\partial Q_F}]$	$[\frac{\partial Q_2}{\partial Q_F}]$	$[\frac{\partial C_{v1}}{\partial Q_F}]$	$[\frac{\partial P_F}{\partial Q_F}]$	$[\frac{\partial P_1}{\partial Q_F}]$	$[\frac{\partial C_{v2}}{\partial Q_F}]$	$[\frac{\partial P_2}{\partial Q_F}]$
+	+	−	−	0	−	0	+
+	+	+	−	0	−	0	−
+	+	0	−	0	−	0	0

Note that the above examples are similar to the examples 1, 3, and 8, respectively, of Section 5.2. In fact, the results of the above examples could be derived from the results of the corresponding examples in Section 5.2 and by using Proposition 3.4. In other words, we can solve any sign analysis of total differentials question in which Q_F is changed by the method of gains using T_1, T_2, and T_3.

The above discussion was for Q_F. We can do the same analysis for another exogenous variable or constant as well (or for an endogenous variable if it makes sense).

5.4. Gains

To begin value analysis, we must first compute the gains of the model for a particular system state.

Let us consider a system state where the values of exogenous variables and constants are given as

$$Q_F=8 \quad C_{v1}=2 \quad C_{v2}=3 \quad P_1=8 \quad P_2=5$$

We substitute these values into the equations

$$F_1=8-Q_1-Q_2=0$$
$$F_2=Q_1-2\sqrt{P_F-8}=0$$
$$F_3=Q_2-3\sqrt{P_F-5}=0$$

and solve for the values of endogenous variables, yielding

$$Q_1=2 \quad Q_2=6 \quad P_F=9$$

Thus the system state is

$$Q_F=8 \quad C_{v1}=2 \quad C_{v2}=3 \quad P_1=8 \quad P_2=5 \quad Q_1=2 \quad Q_2=6 \quad P_F=9 \qquad (5.1)$$

The equations for gains are

$$\frac{\partial F_1}{\partial Q_1}\frac{\partial Q_1}{\partial y}+\frac{\partial F_1}{\partial Q_2}\frac{\partial Q_2}{\partial y}+\frac{\partial F_1}{\partial P_F}\frac{\partial P_F}{\partial y}=-\frac{\partial F_1}{\partial y}$$

$$\frac{\partial F_2}{\partial Q_1}\frac{\partial Q_1}{\partial y}+\frac{\partial F_2}{\partial Q_2}\frac{\partial Q_2}{\partial y}+\frac{\partial F_2}{\partial P_F}\frac{\partial P_F}{\partial y}=-\frac{\partial F_2}{\partial y}$$

$$\frac{\partial F_3}{\partial Q_1}\frac{\partial Q_1}{\partial y}+\frac{\partial F_3}{\partial Q_2}\frac{\partial Q_2}{\partial y}+\frac{\partial F_3}{\partial P_F}\frac{\partial P_F}{\partial y}=-\frac{\partial F_3}{\partial y}$$

where $y \in \{Q_F, C_{v1}, P_1, C_{v2}, P_2\}$. This equation set can be shown in matrix form as

$$J\underline{x}_y=-\underline{f}_y$$

where J is the Jacobian matrix, x_y and f_y are the column vectors

$$x_y = \begin{bmatrix} \frac{\partial Q_1}{\partial y} \\ \frac{\partial Q_2}{\partial y} \\ \frac{\partial P_F}{\partial y} \end{bmatrix} \qquad f_y = \begin{bmatrix} \frac{\partial F_1}{\partial y} \\ \frac{\partial F_2}{\partial y} \\ \frac{\partial F_3}{\partial y} \end{bmatrix}$$

for the exogenous variable y. The partial derivatives, $\frac{\partial F_i}{\partial x}$, $i=1,...,3$, $x\in\{Q_1,Q_2,P_F\}$, are computed as

$$\begin{array}{lll} \frac{\partial F_1}{\partial Q_1}=-1 & \frac{\partial F_1}{\partial Q_2}=-1 & \frac{\partial F_1}{\partial P_F}=0 \\ \frac{\partial F_2}{\partial Q_1}=1 & \frac{\partial F_2}{\partial Q_2}=0 & \frac{\partial F_2}{\partial P_F}=-1 \\ \frac{\partial F_3}{\partial Q_1}=0 & \frac{\partial F_3}{\partial Q_2}=1 & \frac{\partial F_3}{\partial P_F}=-\frac{3}{4} \end{array}$$

which constitute the Jacobian matrix

$$J=\begin{bmatrix} -1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & -3/4 \end{bmatrix}$$

The partial derivatives wrt exogenous variables, $\frac{\partial F_i}{\partial y}$, $i=1,...3$, are computed as

$$\begin{array}{lllll} \frac{\partial F_1}{\partial Q_F}=1 & \frac{\partial F_1}{\partial C_{v1}}=0 & \frac{\partial F_1}{\partial P_1}=0 & \frac{\partial F_1}{\partial C_{v2}}=0 & \frac{\partial F_1}{\partial P_2}=0 \\ \frac{\partial F_2}{\partial Q_F}=0 & \frac{\partial F_2}{\partial C_{v1}}=-1 & \frac{\partial F_2}{\partial P_1}=1 & \frac{\partial F_2}{\partial C_{v2}}=0 & \frac{\partial F_2}{\partial P_2}=0 \\ \frac{\partial F_3}{\partial Q_F}=0 & \frac{\partial F_3}{\partial C_{v1}}=0 & \frac{\partial F_3}{\partial P_1}=0 & \frac{\partial F_3}{\partial C_{v2}}=-2 & \frac{\partial F_3}{\partial P_2}=\frac{3}{4} \end{array}$$

from which the vectors f_y are formed:

$$f_{Q_F}=\begin{bmatrix}1\\0\\0\end{bmatrix} \quad f_{C_{v1}}=\begin{bmatrix}0\\-1\\0\end{bmatrix} \quad f_{P_1}=\begin{bmatrix}0\\1\\0\end{bmatrix} \quad f_{C_{v2}}=\begin{bmatrix}0\\0\\-2\end{bmatrix} \quad f_{P_2}=\begin{bmatrix}0\\0\\3/4\end{bmatrix}$$

Solving for x_y, we get

$$x_{Q_F}=\begin{bmatrix}4/7\\3/7\\4/7\end{bmatrix} \quad x_{C_{v1}}=\begin{bmatrix}3/7\\-3/7\\-4/7\end{bmatrix} \quad x_{P_1}=\begin{bmatrix}-3/7\\3/7\\4/7\end{bmatrix} \quad x_{C_{v2}}=\begin{bmatrix}-8/7\\8/7\\-8/7\end{bmatrix} \quad x_{P_2}=\begin{bmatrix}3/7\\-3/7\\3/7\end{bmatrix}$$

Thus the gains are

$$\frac{\partial Q_1}{\partial Q_F}=\frac{4}{7} \quad \frac{\partial Q_2}{\partial Q_F}=\frac{3}{7} \quad \frac{\partial P_F}{\partial Q_F}=\frac{4}{7}$$

$$\frac{\partial Q_1}{\partial C_{v1}}=\frac{3}{7} \quad \frac{\partial Q_2}{\partial C_{v1}}=-\frac{3}{7} \quad \frac{\partial P_F}{\partial C_{v1}}=-\frac{4}{7}$$

$$\frac{\partial Q_1}{\partial P_1}=-\frac{3}{7} \quad \frac{\partial Q_2}{\partial P_1}=\frac{3}{7} \quad \frac{\partial P_F}{\partial P_1}=\frac{4}{7}$$

$$\frac{\partial Q_1}{\partial C_{v2}}=-\frac{8}{7} \quad \frac{\partial Q_2}{\partial C_{v2}}=\frac{8}{7} \quad \frac{\partial P_F}{\partial C_{v2}}=-\frac{8}{7}$$

$$\frac{\partial Q_1}{\partial P_2}=\frac{3}{7} \quad \frac{\partial Q_2}{\partial P_2}=-\frac{3}{7} \quad \frac{\partial P_F}{\partial P_2}=\frac{3}{7}$$

For example, the gains wrt Q_F say that if Q_F is changed then Q_1 will change $\frac{4}{7}$ times of the change in Q_F, Q_2 will change $\frac{3}{7}$ times of the change in Q_F, and P_F will change $\frac{4}{7}$ times of the change in Q_F, all three being in the same direction as Q_F.

The total differentials of endogenous variables are

$$dQ_1=\frac{\partial Q_1}{\partial Q_F}\ dQ_F+\frac{\partial Q_1}{\partial C_{v1}}\ dC_{v1}+\frac{\partial Q_1}{\partial P_1}\ dP_1+\frac{\partial Q_1}{\partial C_{v2}}\ dC_{v2}+\frac{\partial Q_1}{\partial P_2}\ dP_2$$

$$dQ_2=\frac{\partial Q_2}{\partial Q_F}\ dQ_F+\frac{\partial Q_2}{\partial C_{v1}}\ dC_{v1}+\frac{\partial Q_2}{\partial P_1}\ dP_1+\frac{\partial Q_2}{\partial C_{v2}}\ dC_{v2}+\frac{\partial Q_2}{\partial P_2}\ dP_2$$

$$dP_F=\frac{\partial P_F}{\partial Q_F}\ dQ_F+\frac{\partial P_F}{\partial C_{v1}}\ dC_{v1}+\frac{\partial P_F}{\partial P_1}\ dP_1+\frac{\partial P_F}{\partial C_{v2}}\ dC_{v2}+\frac{\partial P_F}{\partial P_2}\ dP_2$$

By substituting the gains, we arrive at

$$\begin{aligned}
dQ_1&=\frac{4}{7}\ dQ_F+\frac{3}{7}\ dC_{v1}-\frac{3}{7}\ dP_1-\frac{8}{7}\ dC_{v2}+\frac{3}{7}\ dP_2\\
dQ_2&=\frac{3}{7}\ dQ_F-\frac{3}{7}\ dC_{v1}+\frac{3}{7}\ dP_1+\frac{8}{7}\ dC_{v2}-\frac{3}{7}\ dP_2\\
dP_F&=\frac{4}{7}\ dQ_F-\frac{4}{7}\ dC_{v1}+\frac{4}{7}\ dP_1-\frac{8}{7}\ dC_{v2}+\frac{3}{7}\ dP_2
\end{aligned}\qquad(5.2)$$

5.5. Elasticities

Once the gains are computed the elasticities can be found easily. Using (4.12) where $x\in\{Q_1,Q_2,P_F\}$ and $y\in\{Q_F,C_{v1},P_1,C_{v2},P_2\}$, the elasticities are:

$$\begin{array}{lll}
E_{Q_1Q_F}=\frac{16}{7} & E_{Q_2Q_F}=\frac{4}{7} & E_{P_FQ_F}=\frac{32}{63}\\[2ex]
E_{Q_1C_{v1}}=\frac{3}{7} & E_{Q_2C_{v1}}=-\frac{1}{7} & E_{P_FC_{v1}}=-\frac{8}{63}\\[2ex]
E_{Q_1P_1}=-\frac{12}{7} & E_{Q_2P_1}=\frac{4}{7} & E_{P_FP_1}=\frac{32}{63}\\[2ex]
E_{Q_1C_{v2}}=-\frac{12}{7} & E_{Q_2C_{v2}}=\frac{4}{7} & E_{P_FC_{v2}}=-\frac{8}{21}\\[2ex]
E_{Q_1P_2}=\frac{15}{14} & E_{Q_2P_2}=-\frac{5}{14} & E_{P_FP_2}=\frac{5}{21}
\end{array}$$

5.6. Value Analysis

As the gains are computed for the system state given in (5.1), we can begin value analysis. We repeat the equations of total differentials, shown in (5.2), below:

$$\begin{aligned} &dQ_1-\frac{4}{7}\,dQ_F-\frac{3}{7}\,dC_{v1}+\frac{3}{7}\,dP_1+\frac{8}{7}\,dC_{v2}-\frac{3}{7}\,dP_2=0 \\ &dQ_2-\frac{3}{7}\,dQ_F+\frac{3}{7}\,dC_{v1}-\frac{3}{7}\,dP_1-\frac{8}{7}\,dC_{v2}+\frac{3}{7}\,dP_2=0 \\ &dP_F-\frac{4}{7}\,dQ_F+\frac{4}{7}\,dC_{v1}-\frac{4}{7}\,dP_1+\frac{8}{7}\,dC_{v2}-\frac{3}{7}\,dP_2=0 \end{aligned} \qquad (5.3)$$

In order to solve this set uniquely, we need to specify the values of five of the unknowns.

Consider the following examples:

Example 5.6.1. We increase Q_F 1 unit, while keeping other exogenous variables unchanged. How do the endogenous variables change?

Substituting $dQ_F=1, dC_{v1}=dP_1=dC_{v2}=dP_2=0$ into (5.3), we get

$$dQ_1=\frac{4}{7}$$
$$dQ_2=\frac{3}{7}$$
$$dP_F=\frac{4}{7}$$

Q_1 increases $\frac{4}{7}$ times the increase in Q_F, Q_2 increases $\frac{3}{7}$ times the increase in Q_F, and P_F increases $\frac{4}{7}$ times the increase in Q_F.

Example 5.6.2. We increase Q_F and decrease C_{v1}, both in the same amount, while keeping other exogenous variables unchanged. How do the endogenous variables change?

Substituting $dQ_F=1$, $dC_{v1}=-1$, $dP_1=dC_{v2}=dP_2=0$, we get

$$dQ_1=\frac{1}{7}$$
$$dQ_2=\frac{6}{7}$$
$$dP_F=\frac{8}{7}$$

Q_1 increases $\frac{1}{7}$ times the increase in Q_F, Q_2 increases $\frac{6}{7}$ times the increase in Q_F, and P_F increases $\frac{8}{7}$ times the increase in Q_F.

Note that in Section 5.2, the sign analysis of the same question has resulted in a table containing 3 possibilities. For the particular system state and the amounts of perturbations in Q_F and C_{v1}, we arrive at the above solution.

Example 5.6.3. We increase P_1 1 unit while keeping Q_F, C_{v1}, and C_{v2} unchanged. We want P_F not to change. How must we change P_2 to satisfy this condition and how do Q_1 and Q_2 change?

Substituting $dP_1=1$, $dQ_F=dC_{v1}=dC_{v2}=dP_F=0$, we get

$$dP_2=-\frac{4}{3}$$
$$dQ_1=-1$$
$$dQ_2=1$$

P_2 must be decreased $\frac{4}{3}$ times the increase in P_1. Q_1 decreases and Q_2 increases an amount equal to the increase in P_1.

Example 5.6.4. We decrease Q_F 1 unit and increase C_{v1} 2 units, while keeping C_{v2} unchanged. We want Q_1 to decrease 1 unit and P_F not to change. How must P_1 and P_2 be changed to satisfy this condition and how does Q_2 changes?

Substituting $dQ_F=dQ_1=-1$, $dC_{v1}=2$, $dC_{v2}=dP_F=0$, we get

$$dP_1=3$$
$$dP_2=0$$
$$dQ_2=0$$

P_1 must be increased 3 times the increase in Q_F and P_2 must be kept unchanged. As a result, Q_2 does not change.

Example 5.6.5. We want P_F to increase 1 unit and Q_1 and Q_2 remain unchanged. How must exogenous variables be changed in order to satisfy this condition?

Substituting $dQ_1=dQ_2=0$ and $dP_F=1$, there remains 5 unknowns in 3 equations, so the equations have infinite number of solutions. Consider the sign analysis of this question, which was done in Section 5.2 and resulted in Table 36. Suppose that we decided to decrease C_{v1} and C_{v2} both in 1 units. Substituting $dC_{v1}=dC_{v2}=-1$, we arrive at the answer

$$dQ_F = 0$$

$$dP_1 = 0$$

$$dP_2 = -\frac{5}{3}$$

When we decrease C_{v1} and C_{v2} 1 units, decreasing P_2 $\frac{5}{3}$ units and keeping Q_F and P_1 unchanged satisfy the above condition.

VI. IMPLEMENTATION

The techniques introduced in this work are coded as a computer program named as KGMM (Knowledge Generation from Mathematical Models). The program is written using Turbo Pascal v.3.0 under the operating system MSDOS 3.1.

In this section we will describe the program. Appendices D and E contain a sample run session and the listing of the program, respectively.

6.1. Notes About the Program

A model consists of a name, equations, and parameters. Naming of models is necessary for the purpose of reference. Each equation is an array of characters, as will be explained below. For each parameter, there are two items: the name of the parameter and the type of the parameter (exogenous variable, endogenous variable, or constant). A parameter name consists of a letter followed by any combination of letters and digits.

As explained in Section 2, a model can be formed from several submodels, each submodel from sub-submodels, and so on, called assembling. For this purpose the program maintains a model library. Each record of the library is a model. The library is maintained in a file and it can be listed, new models can be added, or existing models can be removed from it at will.

The model being examined by the program at any time is referred to as the *current model*. Current model is the one that is loaded into memory. All the operations are performed on the current model. The current model can be changed as desired by loading another model as the current one.

The bottom line of the screen is reserved for messages. A message may indicate either an error or the successful completion of an operation. Each error message has a number. When an error message is given the operation is terminated and the execution returns to the main menu. Messages will be explained in Appendix B. Also, on the top of the screen, the name of the current model is displayed.

An operation may be terminated by the user by pressing Ctrl-C at any time. In case long outputs are generated, the Ctrl-S key can be used to pause the screen.

As will be explained in Section 6.2, main menu options 1 and 11 require the user to give a model as input. First the name of the model is given. The library is searched to see whether it contains a model with the same name. If so, an error occurs and the operation terminates. Otherwise, the user is asked for the equations. This can be done in one of two ways: When the program is waiting for an equation to be given, the user may either write the equation explicitly or write the name of a library model preceded by the symbol @. In the first case, the equation is parsed and the user is asked for the next equation. In the second case, the assembling process is activated: the model with the given name, call it the submodel, is loaded from the library (if no such model exists, an error occurs), the equations of this submodel are added to the equations of the model, and the names and types of the parameters of the submodel are asked for. These are the names and types that will be used for the model. That is, parameters are renamed as explained in Section 2. If the name of a parameter is not given, it defaults to the name in the submodel. Similarly, the default for the type of a parameter is its type in the submodel. When the assembling is completed, the user is asked for the next equation. After all the equations are given, the types of those parameters whose types are not already given are requested. This completes the input of the model.

An equation is an arithmetic expression formed from operands (parameter names, numbers, and function designators) and operators. Operators fall into four categories, denoted by their order of precedence:

(1) Unary minus (minus with one operand only)

(2) Power operator : ^

(3) Multiplying operators : * and /

(4) Adding operators : + and -.

Sequences of operators of the same precedence are evaluated from left to right. Expressions within parantheses are evaluated first and independently of preceding and succeeding operators.

A function designator is a function identifier followed by an expression enclosed in parantheses. The occurrence of a function designator causes the function with that name to be activated. The functions currently supported by the program are (Num denotes an expression):

Sin(Num)
returns the sine of Num, which is expressed in radians.

Cos(Num)
returns the cosine of Num, which is expressed in radians.

Arctan(Num)
returns the angle, in radians, whose tangent is Num.

Exp(Num)
returns the exponential of Num, i.e. e^{Num}.

Log(Num)
returns the common logarithm of Num.

Ln(Num)
returns the natural logarithm of Num.

Sqr(Num)
returns the square of Num, i.e. Num*Num.

Sqrt(Num)
returns the square root of Num.

As will be explained in Section 6.2, main menu option 3 requires the calculation of signs of expressions. To calculate the sign of an expression, without knowing the values of parameters, we use qualitative arithmetic. Figure 12 shows the tables for operators and functions used in qualitative arithmetic ($\ominus$ denotes unary minus). The operators ^, *, /, +, and - take two expressions; the operator unary minus and all the functions take a single expression. On the upper left-hand corner of the table is the symbol of the

+	+	0	−
+	+	+	?
0	+	0	−
−	?	−	−

−	+	0	−
+	?	+	+
0	−	0	+
−	−	−	?

*	+	0	−
+	+	0	−
0	0	0	0
−	−	0	+

/	+	0	−
+	+	!	−
0	0	!	0
−	−	!	+

^	+	0	−
+	+	+	+
0	0	!	!
−	?	+	?

⊖	+	0	−
	−	0	+

Sin	+	0	−
	?	0	?

Cos	+	0	−
	?	+	?

Exp	+	0	−
	+	+	+

Ln	+	0	−
	?	!	!

Log	+	0	−
	?	!	!

Sqr	+	0	−
	+	0	+

Sqrt	+	0	−
	+	0	!

Arctan	+	0	−
	+	0	−

FIGURE 12. Operator and function results in qualitative arithmetic

operator or the name of the function. For a binary operator

a ⊕ b

where a and b are expressions, and ⊕∈{^,*,/,+,−}, the rows of the table correspond to [a], the columns to [b], and an entry in the i^th row and j^th column, i,j=1,...,3, corresponds to [a⊕b] for [a] is the sign in the i^th row and [b] is the sign in the j^th column. For a unary operator

⊕ (a)

where a is an expression and ⊕ is the unary minus operator or any one of the functions, the columns of the table correspond to [a] and an entry in the i^th column, i=1,...,3, corresponds to [⊕ (a)] for [a] is the sign in the i^th column.

An entry is one of the following:

+ : the result is positive
- : the result is negative
0 : the result is zero
? : the result is indeterminate
! : the result is not defined

The sign of an expression is computed by using these tables and by getting from the user as much information as necessary to determine the sign uniquely. The calculation is done recursively, beginning from the operator/function having the lowest precedence. For a parameter, the sign of the parameter is given by the user. For an operator or a function, tables in Figure 12 are used. This is explained below for each operator and function, where a and b denote expressions, and Sgn (c), where c is an expression, denotes a function that returns a positive number if [c]=+, a negative number if [c]=-, and the number zero if [c]=0.

a+b

[a] and [b] are computed. If one is positive and the other is negative then [a+b] is given. Otherwise [a+b] is [Sgn(a)+Sgn(b)].

a-b

[a] and [b] are computed. If both of them are positive or both are negative then [a-b] is given. Otherwise [a-b] is [Sgn(a)-Sgn(b)].

a*b

[a] is computed. If [a]=0 then [a*b]=0. Otherwise [b] is computed and [a*b] is [Sgn(a)*Sgn(b)]. (We can as well compute [b] first and evaluate [a*b] as 0 if [b]=0, since multiplication is commutative).

a/b

[a] and [b] are computed. If [b]=0 then [a/b] is undefined. Otherwise [a/b] is [Sgn(a)/Sgn(b)].

a^b

[a] is computed. If [a]=+ then [a^b]=+. Otherwise [b] is computed. If [a]=- and [b]∈{+,-} then [a^b] is given. If [a]=- and [b]=0 then [a^b]=+. If [a]=0 and [b]=+ then [a^b]=0. Otherwise the operation is undefined.

-a

[a] is computed. [-a] is [$\bar{a}$].

Sin(a)

[a] is computed. If [a]=0 then [Sin(a)]=0. Otherwise [Sin(a)] is given.

Cos(a)

[a] is computed. If [a]=0 then [Cos(a)]=+. Otherwise [Cos(a)] is given.

Arctan(a)

[a] is computed. [Arctan(a)] is [a].

Exp(a)

[Exp(a)] is +.

Log(a)

[a] is computed. If [a]=+ then [Log(a)] is given. Otherwise the operation is undefined.

Ln(a)

[a] is computed. If [a]=+ then [Ln(a)] is given. Otherwise the operation is undefined.

Sqr(a)

[a] is computed. If [a]=0 then [Sqr(a)]=0. Otherwise [Sqr(a)]=+.

Sqrt(a)

[a] is computed. If [a]∈{+,0} then [Sqrt(a)]=[a]. Otherwise the operation is undefined.

Consider the following example:

$\mathrm{Sqrt}(z_1+\mathrm{Exp}(z_2))*(z_1/10)$

To find $[\mathrm{Sqrt}(z_1+\mathrm{Exp}(z_2))*(z_1/10)]$ we must compute $[\mathrm{Sqrt}(z_1+\mathrm{Exp}(z_2))]$ and $[z_1/10]$ since * is the operator having the lowest precedence. For $[\mathrm{Sqrt}(z_1+\mathrm{Exp}(z_2))]$ we must compute $[z_1+\mathrm{Exp}(z_2)]$. Now we must compute $[z_1]$ and $[\mathrm{Exp}(z_2)]$. Since z_1 is a parameter, $[z_1]$ is given by the user, call it s_1. $[\mathrm{Exp}(z_2)]$ is + regardless of $[z_2]$. Thus $[z_1+\mathrm{Exp}(z_2)]$ is found from the table of addition with one sign s_1 and the other +. Call the result s_2. Then $[\mathrm{Sqrt}(z_1+\mathrm{Exp}(z_2))]=s_2$ unless $s_2=-$, in which case the operation is undefined. For $[z_1/10]$ we must compute $[z_1]$ and $[10]$. $[z_1]$ is already given as s_1. $[10]$ evaluates to +. Then $[z_1/10]$ is found from the table of division with the first sign s_1 and the second one +, call it s_3. Finally we use the table for multiplication, with one sign s_2 and the other s_3 for the result.

6.2. Options

The main menu is shown in Figure 13. To execute an option, enter the number shown on the left-hand side and then press <Return>.

The first option is used to input a mathematical model as the current model. When this option is selected, the current model, if any, is deleted from memory and the model given by the user as explained in Section 6.1 becomes the current model. If the number of endogenous variables is not equal to the number of equations, then an error occurs. Note that if an error is encountered during the process, no current model exists in memory.

```
Analysis of Algebraic Mathematical Models  ver 1.00
Main__Menu
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit
```

FIGURE 13. Program main menu

Option 2 is used to load a model from library to the memory as the current model. When this option is selected, the current model, if any, is deleted from memory and a model of the library becomes the current model. The user is asked for the name of a library model. The library is searched and the model with the given name is loaded to memory and becomes the current model provided that the model exists in the library. Otherwise an error occurs and the option is terminated (but the current model, if any, continues its existence).

Option 3 is used to reduce the Information matrix, I, of the current model into a sign matrix. This matrix is used to form the sign tables of the modeling equations. As explained in Section 6.1, in order to reduce this matrix into a sign matrix, we must compute the sign of each entry in the matrix. For this purpose the user is asked for the necessary information about the expressions. This process must precede the sign analysis (options 4 and 5). It restricts the system space of the current model. Once this process is completed, all the sign analysis results will be valid for this restricted system space until this option is selected again and the system space is changed.

If no current model exists when this option is selected, an error occurs.

Option 4 is used to perform sign analysis of total differentials for the current model. The user is asked for the direction of change of each parameter as in the following format:

+ : the parameter is increased
0 : the parameter is kept unchanged
– : the parameter is decreased
? : the direction of change of the parameter is unknown and will be determined by the analysis

The perturbation formula is formed from the parameters given in the first three cases, and the analysis results the directions of change of the parameters given in the last case. Finally the number of possibilities found is displayed. The analysis uses the system space given by option 3.

If no current model exists or if option 3 has not been selected before this option then an error occurs.

Option 5 is used to perform sign analysis of gains for the current model. The user is asked for the name of the exogenous variable or constant, say z, whose value is to be changed. Then the directions of change of other exogenous variables, relative to z, are given in the following format:

+ : the parameter is changed in the same direction as z
0 : the parameter is kept unchanged
- : the parameter is changed in the opposite direction of z

The result is the directions of change of the endogenous variables relative to z. The analysis uses the system space given by option 3.

If no current model exists or if option 3 has not been selected before this option then an error occurs.

Option 6 is used to give the values of the parameters, i.e. the system state of the current model, before performing value analysis. The values of all the parameters are given. Then the user is asked whether he/she wants the equations to be solved for the values of the endogenous variables. If so, the values given for the endogenous variables are treated as the initial values and the equations are solved by using Newton-Raphson method. This iteration method is explained in Appendix A. If the equations converge, then the values of the endogenous variables are found for the given set of the values of the exogenous variables and constants. If the equations do not converge, then an error occurs and the parameter values become undetermined. In this case, the user may repeat the process with different initial values. If the user does not want the equations to be solved, then the values given are treated as the system state, i.e. they satisfy the equations.

Once this process is completed, all the value analysis results will be valid for this system state until this option is selected again and the system state is changed.

If no current model exists when this option is selected, an error occurs.

Option 7 is used to perform value analysis on the current model. The user is asked for the directions and magnitudes of change of the parameters in the following format:

+ : the parameter is increased
0 : the parameter is kept unchanged
- : the parameter is decreased
? : the direction of change of the parameter is unknown and will be determined by the analysis

Magnitude is needed only if the direction is + or -. The number of unknown parameters must be equal to the number of equations. The result is the magnitudes of the unknown parameters. The analysis uses the system state given by option 6.

If no current model exists or if option 6 has not been selected before this option then an error occurs.

Option 8 is used to display the gains of the current model for the system state given by option 6. The user is asked for the name of the exogenous variable or constant whose gains are to be displayed. If no parameter name is specified then the gains wrt all exogenous variables will be outputted.

If no current model exists or if option 6 has not been selected before this option then an error occurs.

Option 9 is used to display the elasticities of the current model for the system state given by option 6. The user is asked for the name of the exogenous variable or constant whose elasticities are to be displayed. If no parameter name is specified then the elasticities wrt all exogenous variables will be outputted.

If no current model exists or if option 6 has not been selected before this option then an error occurs.

Option 10 is used to list the models in the library.

Option 11 is used to insert a new model into the library. The model is given by the user as explained in Section 6.1. Note that the current model, if any, is not affected by this process.

Option 12 is used to delete a model from the library. The name of the model is given. Then the library is searched to see whether the model with the given name exists. If so, the model is removed from the library. If no such model exists, then an error occurs. Note that the current model, if any, is not affected by this process.

Option 13 is used to insert the current model into the library. If the model is already in the library, then an error occurs. This process is useful when one wants to continue the analysis of a model at a latter time. He/she inserts the model into the library and loads it again from the library by using option 2 at another time and analyses it.

Option 14 is used to terminate the execution of the program and return to DOS.

VII. CONCLUSION

In this work we have introduced several techniques to generate causal relations from mathematical models of the form of algebraic equations. The techniques are based on the use of partial derivatives to analyse the signs of total differentials of parameters, and to compute the gains and the elasticities of parameters, which are then converted into causal relations between parameters.

Our work is a step towards the automation of analysis of systems. System analysis is a broad concept concerning many scientific disciplines; and we have attempted to formalize only one aspect of the problem, which is to extract causal relations regardless of the domain. There remains several questions which require additional work. Below we discuss briefly possible future extensions of our work.

Causal relations are expressed in terms of the parameters of the original mathematical model. On the other hand the domain of the mathematical model may not be so suitable for use directly in the reasoning process. Transformation of causal relations from the domain of the mathematical models to a domain used in commonsense reasoning is one of the most challenging areas requiring future work. One strategy is to make some simplifying assumptions, possibly by using domain-dependent knowledge, on the causal relations for typical cases and extreme cases of a system and to express these assumptions as either preconditions for the causal relations or as probabilities.

Another area to work on is to extend the sign analysis and value analysis techniques to other types of mathematical models. In that sense models expressed in terms of differential equations, difference equations, and mix of algebraic, differential and difference equations can be considered. Some new techniques may as well emerge for these types of models.

In their work [2], de Kleer and Brown discuss two types of behavior: Intrastate behavior and interstate behavior. The work in this thesis has considered only the first one of these. The equations of a model are in fact valid for a certain region of the values the parameters may take on. As the parameter values go outside of this region for some reason, the form of the equations change and we continue the analysis with a new set of equations. The work can be extended in this direction to take into account the transformation between states by using methods that are similar to those of other researchers.

In Section 3.2, we have shown that sign analysis technique can generate spurious solutions, i.e. solutions that are not actual for the system. Since a mathematical model, as in the form discussed in this thesis, contains complete information about its parameters, it must be possible to prevent spurious solutions. We think that the most plausible way to do this is to extend the sign tables in the sense that each element of a sign table will have a magnitude in addition to the sign of the partial derivative. In fact, for sign analysis, the equations in a model need not be quantitative in form. For instance, the constraint equations suggested in [5] include certain relationships between parameters such as monotonically increasing and monotonically decreasing. In such a case the ambiguity in the solutions is unavoidable. To prevent this the model must contain sufficient quantitative information.

One of the attractive application areas of the techniques is the diagnosis of faults. Possible behaviors of a system can be generated, particularly to warn of surprising or disastrous events. Also the techniques can be used for the generation and synthesis of fault trees. A fault tree is used to explain the possible causes of an undesirable event associated with the system [15]. The construction of a fault tree requires a structure, such as a directed graph, that represents the causal relations between the parameters in the system, where our techniques may prove useful.

An expanded form of value analysis is interval analysis. Instead of assigning a single value to a parameter, we can represent the typical values the parameter may take on by an interval. In that sense interval analysis is in between sign analysis (analysis on system space) and value analysis (analysis for a particular system state). The same principles used in value analysis can be applied to interval analysis. However, as is well-known, interval arithmetic is expensive and ambiguous (e.g. [16]). For example, solving a set of simultaneous linear equations by using different methods may give different results. Also obtaining results that are precise enough for the analysis to make sense may require excessive computation time.

The techniques presented in this work may seem expensive in terms of computation time. However this is not exactly the case. The most time-consuming one is the sign analysis. This becomes clear if we consider that the sign table of a medium-size function contains thousands of rows and the sign analysis of a question requires the cartesian product (for join) of several such functions. This process far exceeds the memory and time limits. For instance, in the program described in Section 6, no sign tables are created and the join operation is done implicitly. So the computation time is reduced to a reasonable limit. For example, each of the results of Appendix D is obtained in a few seconds on an 8 Mhz computer.

Finally we want to point out that the key to the techniques introduced in this work is the use of the simultaneity in the equations. We think that processing all the equations at a time is more efficient than processing them one by one, because, in the latter case, while generating a solution for an equation, the interrelations between that equation and other equations are overlooked, which may cause an unreasonable solution for the system and force us to backtrack.

APPENDIX A. NEWTON-RAPHSON METHOD

This appendix describes the Newton-Raphson (e.g. [17]) method as it is used in the program for solving a set of simultaneous nonlinear equations.

Consider a system modeled by the following functions

$$\begin{aligned} &F_1(x_1,\ldots,x_n,y_1,\ldots,y_m)=0 \\ &\vdots \\ &F_n(x_1,\ldots,x_n,y_1,\ldots,y_m)=0 \end{aligned}$$

where $x_1,\ldots,x_n$ represent endogenous variables, and $y_1,\ldots,y_m$ represent exogenous variables and constants.

The notation $\alpha^{(k)}$ denotes the value of α at the kth iteration, where α may be a matrix, a column vector, an endogenous variable, a partial derivative, or a function.

The Newton-Raphson method solves a set of simultaneous nonlinear equations, shown above, for the values of the endogenous variables given the values of the exogenous variables and constants and the initial values of the endogenous variables, $x_1^{(0)},\ldots,x_n^{(0)}$.

The iteration formula is

$$\begin{bmatrix} \frac{\partial F_1}{\partial x_1} \cdots \frac{\partial F_1}{\partial x_n} \\ \vdots \quad\quad \vdots \\ \frac{\partial F_n}{\partial x_1} \cdots \frac{\partial F_n}{\partial x_n} \end{bmatrix}^{(k)} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}^{(k+1)} = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} \cdots \frac{\partial F_1}{\partial x_n} \\ \vdots \quad\quad \vdots \\ \frac{\partial F_n}{\partial x_1} \cdots \frac{\partial F_n}{\partial x_n} \end{bmatrix}^{(k)} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}^{(k)} - \begin{bmatrix} F_1 \\ \vdots \\ F_n \end{bmatrix}^{(k)}$$

which can be expressed in matrix form as

$$J^{(k)}\mathbf{x}^{(k+1)}=J^{(k)}\mathbf{x}^{(k)}-\mathbf{f}^{(k)}$$

where J is the Jacobian matrix, $\mathbf{x}$ is the column vector of endogenous variables, and $\mathbf{f}$ is the column vector of function values.

This is a set of n simultaneous linear equations and can be solved for the n unknowns $x_1^{(k+1)},\ldots,x_n^{(k+1)}$, i.e. $\mathbf{x}^{(k+1)}$, by any method for solving simultaneous linear equations.

The iteration continues until either the solution converges or it is understood that the solution will not converge. It is accepted by the program that the solution converges if the number of iterations is equal to or less than 10. Since Newton-Raphson provides fast convergence, usually in three or four steps, it is almost impossible for the solution to converge if convergence has not already been established within 10 steps.

The convergence criteria used by the program is

$$\sqrt{\sum_{i=1}^{n}(x_i^{(k+1)}-x_i^{(k)})^2} < T \qquad \text{and} \qquad \sqrt{\sum_{i=1}^{n}(F_i)^2} < T$$

where T indicates the tolerance, which is equal to $1*10^{-6}$.

APPENDIX B. PROGRAM MESSAGES

The following is a listing of error messages the user may get during the execution of the program. Many error messages are self-explanatory, but if one feels to get some explanation, he/she can refer to the following listing via the error number.

An error causes the active process to terminate. The user is required to press the Esc key and then the execution returns to the main menu.

1 : unexpected end of source (Position=xx)

The equation cannot end the way it does. It must contain some other operands and/or operators. The number xx specifies the position of the error relative to the beginning of the equation string.

2 : unexpected character in equation (Position=xx)

An unexpected character is encountered while expecting a parameter name, a number, an operator, a function identifier, (, or). The number xx specifies the position of the error relative to the beginning of the equation string.

3 : improper equation syntax (Position=xx)

The syntax of the equation is wrong. Check the syntax to agree with the rules of ordinary arithmetic expressions. The number xx specifies the position of the error relative to the beginning of the equation string.

4 : unmatched) (Position=xx)

The right paranthesis does not have a corresponding left paranthesis (. The number xx specifies the position of the error relative to the beginning of the equation string.

5 :) expected (Position=xx)

The number of left parantheses (is greater than the number of right parantheses). The number xx specifies the position of the error relative to the beginning of the equation string.

6 : sign of Information matrix entry cannot be zero (Eqn:xx,Par:yy)

During the process of reducing the Information matrix, I, into a sign matrix, the signs of all the entries of the I matrix (except those entries for which the parameter does not appear in the equation) must be either positive or negative. However, the sign of the indicated entry is computed as zero for the given signs of the expressions. The number xx specifies the number of the equation and the symbol yy specifies the name of the parameter.

7 : # of equations must be equal to # of endogenous vars

In order to be able to perform value analysis on a model, the number of equations must be equal to the number of endogenous variables.

8 : cannot execute selection before a model is given

Before this option, load a current model by using main menu options 1 or 2.

9 : cannot execute selection before sign of I matrix given

The Information matrix, I, of the current model must be reduced into a sign matrix by using main menu option 3 before sign analysis.

10 : cannot execute selection before par values are given

The parameter values, i.e. the system state, must be given by using main menu option 6 before this option.

11 : equations do not converge for the given initial values

The program cannot find the values of the endogenous variables by solving the equations for the given values of the exogenous variables/constants and the initial values of the endogenous variables. Either change the initial values and try again or solve the equations outside the program.

12 : singular matrix encountered in analysis
While solving a set of simultaneous linear equations, a singular matrix is encountered which means that the set cannot be solved uniquely.

13 : error in division operator
Division by zero attempted, the operation is undefined.

14 : error in power operator
The power operation, a^b, is undefined for the values of the numbers a and b.

15 : error in ln operator
The argument of an ln operator is found to be negative or zero.

16 : error in log operator
The argument of a log operator is found to be negative or zero.

17 : error in sqrt operator
The argument of a sqrt operator is found to be negative.

18 : model does not exist
The library does not contain a model with the given name.

19 : model already exists
The library already contains a model with the given name.

20 : invalid selection
No option with the given number in the main menu.

21 : number of equations cannot be greater than 5
Number of equations in a model is limited by 5 (Appendix C).

22 : number of parameters cannot be greater than 20
Number of parameters in a model is limited by 20 (Appendix C).

23 : number of parameters is zero
The model does not contain any parameter.

24 : no such parameter in the current model
The model does not contain a parameter with the given name.

25 : parameter must be an exogenous variable or a constant
This is necessary for value analysis, gains, and elasticities.

26 : # of unknowns must be equal to # of equations
This condition is needed to obtain a unique solution in value analysis.

50 : user break, command aborted
This is actually not an error. It indicates that the process is aborted by the user by pressing Ctrl-C.

In addition to these error messages, there are messages that indicate the successful completion of a process. These are listed below. Each message corresponds to one of the menu options.

When a message is given, the user is required to press the Esc key, after which the main menu is displayed.

Model given by user is loaded as the current model.
Model from library is loaded as the current model.
Signs of I matrix entries are initialized.
Sign analysis of total differentials completed.
Sign analysis of gains completed.
Parameter values are initialized.
Value analysis completed.
Listing of gains completed.
Listing of elasticities completed.
Listing completed.
Current model is inserted to the library.
Model is deleted from library.
Model is inserted to the library.

APPENDIX C. SPECIFICATIONS

The program can be run on IBM PC, XT, AT and compatibles; either on the hard disk or on one of the floopy diskets. It requires approximately 200 KB of memory.

The program disket contains the following files:

KGMM.COM
This file includes the executable code of the program.

MODELLIB
This file contains the models in the model library.

KGMM.PAS
This is the program source file. It is heavily commanded for ease of understanding.

INCLUDE.PAS
This is the iclude file containing additional subroutines.

Among these four files, only the first two need to reside on the disket. If the file MODELLIB is absent, an empty library file will be created.

To execute the program, enter the command

```
KGMM
```

at the terminal. The program will log on and the main menu will be displayed.

The program has some limitations imposed by the program structure. Some of these are due to the memory management method used by the compiler. The program is coded as a prototype for the techniques introduced in this work and it is not intended to analyse large systems. Its main aim is to demonstrate that the techniques work and produce useful results. A more complete implementation can be written by using better programming tools.

The limitations are as follows:

The maximum number of equations in a model : 5
The maximum number of parameters in a model : 20
The maximum length of a parameter name : 20
The maximum length of a model name : 20
The maximum length of an equation : 70
The maximum number of models in the library : 32767.

APPENDIX D. A SAMPLE RUN

In this appendix, we give a sample execution of the program on the example model of Section 5. The analysis will be similar to the one performed in that section.

Before the execution of the program, the library is assumed to be empty.

```
C>KGMM


Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                           Current model :
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 11

Insert to library                                   Current model :
Add a new model to the library
------------------------------------------------------------------------------
Model name : PUMP

Equations
---------
Eqn : QIN-QOUT1-QOUT2
Eqn :
Parameter types (1:Endo  2:Exo  3:Constant)
---------------
QIN        --> 2
QOUT1      --> 1
QOUT2      --> 1

------------------------------------------------------------------------------
model is inserted to the library

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                           Current model :
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 11

Insert to library                                   Current model :
Add a new model to the library
------------------------------------------------------------------------------
Model name : VALVE

Equations
---------
```

```
Eqn : Q-CV*SQRT(P1-P2)
Eqn :
Parameter types (1:Endo  2:Exo  3:Constant)
---------------
Q          --> 1
CV         --> 3
P1         --> 2
P2         --> 2

--------------------------------------------------------------------------------
model is inserted to the library

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                                  Current model :
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 10

List library                                               Current model :
Models in the library are listed
--------------------------------------------------------------------------------
Model no   : 1
Model name : PUMP
Equations
---------
 1. QIN-QOUT1-QOUT2
Parameters (Name & Type)
----------
 1. QIN        Exo  var
 2. QOUT1      Endo var
 3. QOUT2      Endo var

Model no   : 2
Model name : VALVE
Equations
---------
 1. Q-CV*SQRT(P1-P2)
Parameters (Name & Type)
----------
 1. Q          Endo var
 2. CV         Constant
 3. P1         Exo  var
 4. P2         Exo var

listing completed
--------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                                  Current model :
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
```

```
 6.  Parameter values
 7.  Value analysis
 8.  Gains
 9.  Elasticities
 10. List library
 11. Insert to library
 12. Delete from library
 13. Save current model
 14. Exit

 Choice : 1

 New model (from user)                              Current model :
 A new model is given by the user and becomes the current model
 ----------------------------------------------------------------------------
 Model name : MODEL1

 Equations
 ---------
 Eqn : @PUMP

 Model no   : 1
 Model name : PUMP
 Equations
 ---------
  1. QIN-QOUT1-QOUT2
 Parameters (Name & Type)
 ----------
  1. QIN        Exo  var
  2. QOUT1      Endo var
  3. QOUT2      Endo var

 QIN        --> QF
 Type (0:Default,1:Endo,2:Exo,3:Constant) : 0
 QOUT1      --> Q1
 Type (0:Default,1:Endo,2:Exo,3:Constant) : 0
 QOUT2      --> Q2
 Type (0:Default,1:Endo,2:Exo,3:Constant) : 0

 Eqn : @VALVE

 Model no   : 2
 Model name : VALVE
 Equations
 ---------
  1. Q-CV*SQRT(P1-P2)
 Parameters (Name & Type)
 ----------
  1. Q          Endo var
  2. CV         Constant
  3. P1         Exo  var
  4. P2         Exo var

 Q          --> Q1
 CV         --> CV1
 Type (0:Default,1:Endo,2:Exo,3:Constant) : 0
 P1         --> PF
 Type (0:Default,1:Endo,2:Exo,3:Constant) : 1
 P2         --> P1
 Type (0:Default,1:Endo,2:Exo,3:Constant) : 0

 Eqn : @VALVE

 Model no   : 2
 Model name : VALVE
 Equations
 ---------
  1. Q-CV*SQRT(P1-P2)
 Parameters (Name & Type)
```

```
----------
 1. Q         Endo var
 2. CV        Constant
 3. P1        Exo  var
 4. P2        Exo var

Q          --> Q2
CV         --> CV2
Type (0:Default,1:Endo,2:Exo,3:Constant) : 0
P1         --> PF
P2         -->
Type (0:Default,1:Endo,2:Exo,3:Constant) : 0

Eqn :

Parameter types (1:Endo  2:Exo  3:Constant)
---------------

model given by user is loaded as the current model
------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                             Current model : MODEL1
---------

1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 3

Sign of Information matrix
Signs of pars & par expressions given to reduce I matrix into a sign matrix
------------------------------------------------------------------------------
Sign of PF (- 0 +) : +
Sign of P1 (- 0 +) : +
Sign of (PF-P1) (- 0 +) : +
Sign of CV1 (- 0 +) : +
Sign of CV2 (- 0 +) : +
Sign of P2 (- 0 +) : +
Sign of (PF-P2) (- 0 +) : +

signs of I matrix entries are initialized
------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                             Current model : MODEL1
---------

1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
```

```
12. Delete from library
13. Save current model
14. Exit

Choice : 4

Sign analysis of total differentials
Sign analysis of tot difs on current model, signs of ? pars are determined
-------------------------------------------------------------------------------
QF         (- 0 + ?) : +
Q1         (- 0 + ?) : ?
Q2         (- 0 + ?) : ?
CV1        (- 0 + ?) : 0
PF         (- 0 + ?) : ?
P1         (- 0 + ?) : 0
CV2        (- 0 + ?) : 0
P2         (- 0 + ?) : 0

Possibilities
-------------
Q1 = +   Q2 = +   PF = +

Possibilities found : 1

sign analysis of total differentials completed
-------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                          Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 4

Sign analysis of total differentials
Sign analysis of tot difs on current model, signs of ? pars are determined
-------------------------------------------------------------------------------
QF         (- 0 + ?) : 0
Q1         (- 0 + ?) : ?
Q2         (- 0 + ?) : ?
CV1        (- 0 + ?) : -
PF         (- 0 + ?) : ?
P1         (- 0 + ?) : 0
CV2        (- 0 + ?) : 0
P2         (- 0 + ?) : 0

Possibilities
-------------
Q1 = -   Q2 = +   PF = +

Possibilities found : 1

sign analysis of total differentials completed
-------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
```

```
Main Menu                                                 Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 4

Sign analysis of total differentials
Sign analysis of tot difs on current model, signs of ? pars are determined
-----------------------------------------------------------------------------
QF         (- 0 + ?) : +
Q1         (- 0 + ?) : ?
Q2         (- 0 + ?) : ?
CV1        (- 0 + ?) : -
PF         (- 0 + ?) : ?
P1         (- 0 + ?) : 0
CV2        (- 0 + ?) : 0
P2         (- 0 + ?) : 0

Possibilities
-------------
Q1 = -   Q2 = +   PF = +
Q1 = 0   Q2 = +   PF = +
Q1 = +   Q2 = +   PF = +

Possibilities found : 3

sign analysis of total differentials completed
-----------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                                 Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 4

Sign analysis of total differentials
Sign analysis of tot difs on current model, signs of ? pars are determined
-----------------------------------------------------------------------------
QF         (- 0 + ?) : 0
Q1         (- 0 + ?) : +
Q2         (- 0 + ?) : ?
CV1        (- 0 + ?) : 0
```

```
PF          (- 0 + ?) : ?
P1          (- 0 + ?) : 0
CV2         (  0 + ?) : 0
P2          (- 0 + ?) : 0

Inconsistent, no possibilities

sign analysis of total differentials completed
-------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                            Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 4

Sign analysis of total differentials
Sign analysis of tot difs on current model, signs of ? pars are determined
-------------------------------------------------------------------------------
 QF         (- 0 + ?) : +
Q1          (- 0 + ?) : +
Q2          (- 0 + ?) : ?
CV1         (- 0 + ?) : 0
PF          (- 0 + ?) : ?
P1          (- 0 + ?) : 0
CV2         (- 0 + ?) : 0
P2          (- 0 + ?) : 0

Possibilities
-------------
Q2 = +   PF = +

Possibilities found : 1

sign analysis of total differentials completed
-------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                            Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit
```

```
Choice : 4

Sign analysis of total differentials
Sign analysis of tot difs on current model, signs of ? pars are determined
--------------------------------------------------------------------------------
QF          (- 0 + ?) : ?
Q1          (- 0 + ?) : 0
Q2          (- 0 + ?) : 0
CV1         (- 0 + ?) : ?
PF          (- 0 + ?) : +
P1          (- 0 + ?) : ?
CV2         (- 0 + ?) : ?
P2          (- 0 + ?) : ?

Possibilities
-------------
QF = 0   CV1 = -   P1 = +   CV2 = -   P2 = +
QF = 0   CV1 = -   P1 = +   CV2 = -   P2 = 0
QF = 0   CV1 = -   P1 = +   CV2 = -   P2 = -
QF = 0   CV1 = -   P1 = +   CV2 = 0   P2 = +
QF = 0   CV1 = -   P1 = +   CV2 = +   P2 = +
QF = 0   CV1 = -   P1 = 0   CV2 = -   P2 = +
QF = 0   CV1 = -   P1 = 0   CV2 = -   P2 = 0
QF = 0   CV1 = -   P1 = 0   CV2 = -   P2 = -
QF = 0   CV1 = -   P1 = 0   CV2 = 0   P2 = +
QF = 0   CV1 = -   P1 = 0   CV2 = +   P2 = +
QF = 0   CV1 = -   P1 = -   CV2 = -   P2 = +
QF = 0   CV1 = -   P1 = -   CV2 = -   P2 = 0
QF = 0   CV1 = -   P1 = -   CV2 = -   P2 = -
QF = 0   CV1 = -   P1 = -   CV2 = 0   P2 = +
QF = 0   CV1 = -   P1 = -   CV2 = +   P2 = +
QF = 0   CV1 = 0   P1 = +   CV2 = -   P2 = +
QF = 0   CV1 = 0   P1 = +   CV2 = -   P2 = 0
QF = 0   CV1 = 0   P1 = +   CV2 = -   P2 = -
QF = 0   CV1 = 0   P1 = +   CV2 = 0   P2 = +
QF = 0   CV1 = 0   P1 = +   CV2 = +   P2 = +
QF = 0   CV1 = +   P1 = +   CV2 = -   P2 = +
QF = 0   CV1 = +   P1 = +   CV2 = -   P2 = 0
QF = 0   CV1 = +   P1 = +   CV2 = -   P2 = -
QF = 0   CV1 = +   P1 = +   CV2 = 0   P2 = +
QF = 0   CV1 = +   P1 = +   CV2 = +   P2 = +

Possibilities found : 25

sign analysis of total differentials completed
--------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                          Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 4

Sign analysis of total differentials
Sign analysis of tot difs on current model, signs of ? pars are determined
```

```
--------------------------------------------------------------------------------
QF        (- 0 + ?) : 0
Q1        (- 0 + ?) : ?
Q2        (- 0 + ?) : ?
CV1       (- 0 + ?) : 0
PF        (- 0 + ?) : 0
P1        (- 0 + ?) : +
CV2       (- 0 + ?) : 0
P2        (- 0 + ?) : ?

Possibilities
-------------
Q1 = -   Q2 = +   P2 = -

Possibilities found : 1

sign analysis of total differentials completed
--------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                               Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 4

Sign analysis of total differentials
Sign analysis of tot difs on current model, signs of ? pars are determined
--------------------------------------------------------------------------------
QF        (- 0 + ?) : -
Q1        (- 0 + ?) : -
Q2        (- 0 + ?) : ?
CV1       (- 0 + ?) : +
PF        (- 0 + ?) : 0
P1        (- 0 + ?) : ?
CV2       (- 0 + ?) : 0
P2        (- 0 + ?) : ?

Possibilities
-------------
Q2 = -   P1 = +   P2 = +
Q2 = 0   P1 = +   P2 = 0
Q2 = +   P1 = +   P2 = -

Possibilities found : 3

sign analysis of total differentials completed
--------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                               Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
```

```
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 5

Sign analysis of gains
Sign analysis of gains on current model,gains of endos wrt a exo/cst determined
--------------------------------------------------------------------------------
Name of exo var/constant : QF

CV1        (- 0 +) : 0
P1         (- 0 +) : 0
CV2        (- 0 +) : 0
P2         (- 0 +) : 0

Possibilities
-------------
Q1 = +   Q2 = +   PF = +

Possibilities found : 1

sign analysis of gains completed
--------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                                Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 5

Sign analysis of gains
Sign analysis of gains on current model,gains of endos wrt a exo/cst determined
--------------------------------------------------------------------------------
Name of exo var/constant : QF

CV1        (- 0 +) : -
P1         (- 0 +) : 0
CV2        (- 0 +) : 0
P2         (- 0 +) : 0

Possibilities
-------------
Q1 = -   Q2 = +   PF = +
Q1 = 0   Q2 = +   PF = +
Q1 = +   Q2 = +   PF = +

Possibilities found : 3
```

```
sign analysis of gains completed
--------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                          Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 6

Parameter values
Par values given by the user & equations solved, if desired, for endo vars
--------------------------------------------------------------------------------
QF         = 8
Q1         = 1
Q2         = 10
CV1        = 2
PF         = 10
P1         = 8
CV2        = 3
P2         = 5

Will the equations be solved for endogenous variables ? (Y/N) Y

Q1         =   2.00
Q2         =   6.00
PF         =   9.00

parameter values are initialized
--------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                          Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 8

Gains
Display gains for a single exo var/constant or for all exo vars/constants
--------------------------------------------------------------------------------
Name of exo var/constant :
```

```
Gains for QF
------------
Q1          :   0.57
Q2          :   0.43
PF          :   0.57
Gains for CV1
-------------
Q1          :   0.43
Q2          :  -0.43
PF          :  -0.57
Gains for P1
------------
Q1          :  -0.43
Q2          :   0.43
PF          :   0.57
Gains for CV2
-------------
Q1          :  -1.14
Q2          :   1.14
PF          :  -1.14
Gains for P2
------------
Q1          :   0.43
Q2          :  -0.43
PF          :   0.43

listing of gains completed
--------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                                  Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 9

Elasticities
Display elasticities for a single exo var/const or for all exo vars/constants
--------------------------------------------------------------------------------
Name of exo var/constant :

Gains for QF
------------
Q1          :   2.29
Q2          :   0.57
PF          :   0.51
Gains for CV1
-------------
Q1          :   0.43
Q2          :  -0.14
PF          :  -0.13
Gains for P1
------------
Q1          :  -1.71
Q2          :   0.57
```

```
PF          :  0.51
Gains for CV2
-------------
Q1          : -1.71
Q2          :  0.57
PF          : -0.38
Gains for P2
------------
Q1          :  1.07
Q2          : -0.36
PF          :  0.24

listing of elasticities completed
-----------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                                   Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 7

Value analysis
Value analysis of tot difs on current model, magnitudes of ? pars determined
-----------------------------------------------------------------------------
             Sign (- 0 + ?)  Magnitude
             --------------  ---------
QF          :      +              1
Q1          :      ?
Q2          :      ?
CV1         :      0
PF          :      ?
P1          :      0
CV2         :      0
P2          :      0

Results
-------
Q1          =  0.57
Q2          =  0.43
PF          =  0.57

value analysis completed
-----------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                                   Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
```

```
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 7

Value analysis
Value analysis of tot difs on current model, magnitudes of ? pars determined
-------------------------------------------------------------------------------
             Sign (- 0 + ?)  Magnitude
             --------------  ---------
QF          :      +             1
Q1          :      ?
Q2          :      ?
CV1         :      -             1
PF          :      ?
P1          :      0
CV2         :      0
P2          :      0

Results
-------
Q1          =   0.14
Q2          =   0.86
PF          =   1.14

value analysis completed
-------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                              Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 7

Value analysis
Value analysis of tot difs on current model, magnitudes of ? pars determined
-------------------------------------------------------------------------------
             Sign (- 0 + ?)  Magnitude
             --------------  ---------
QF          :      0
Q1          :      ?
Q2          :      ?
CV1         :      0
PF          :      0
P1          :      +             1
CV2         :      0
P2          :      ?

Results
-------
Q1          =  -1.00
```

```
Q2         =  1.00
P2         =  1.33

value analysis completed
--------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                            Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 7

Value analysis
Value analysis of tot difs on current model, magnitudes of ? pars determined
--------------------------------------------------------------------------------
             Sign (- 0 + ?)  Magnitude
             --------------  ---------
QF         :       -             1
Q1         :       -             1
Q2         :       ?
CV1        :       +             2
PF         :       0
P1         :       ?
CV2        :       0
P2         :       ?

Results
-------
Q2         =  0.00
P1         =  3.00
P2         =  0.00

value analysis completed
--------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                            Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 7
```

```
Value analysis
Value analysis of tot difs on current model, magnitudes of ? pars determined
-------------------------------------------------------------------------------
              Sign (- 0 + ?)  Magnitude
              --------------  ---------
QF          :      ?
Q1          :      0
Q2          :      0
CV1         :      -              1
PF          :      +              1
P1          :      ?
CV2         :      -              1
P2          :      ?

Results
-------
QF          =   0.00
P1          =   0.00
P2          =  -1.67

value analysis completed
-------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                          Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 13

Save current model
Add the current model to the library
-------------------------------------------------------------------------------

current model is inserted to the library
-------------------------------------------------------------------------------

Analysis of Algebraic Mathematical Models  ver.1.00
Main Menu                                          Current model : MODEL1
---------
1.  New model (from user)
2.  New model (from library)
3.  Sign of Information matrix
4.  Sign analysis of total differentials
5.  Sign analysis of gains
6.  Parameter values
7.  Value analysis
8.  Gains
9.  Elasticities
10. List library
11. Insert to library
12. Delete from library
13. Save current model
14. Exit

Choice : 14

C>
```

APPENDIX E. PROGRAM LISTING

```
{$C-}
{$V-}

Program KnowledgeGenerationFromMathematicalModels (Input, Output) ;

Const
  MaxEquation         =  5 ;                              { max # of equations in a model }
  MaxParameter        = 20 ;                             { max # of parameters in a model }
  ParameterNameLen    = 10 ;                        { # of characters in a parameter's name }
  MaxPostfixEntry     = 30 ;                     { max # of entries in a postfix expression }
  ModelNameLen        = 10 ;                            { # of characters in a model's name }

  ModelLibraryFileName = 'MODELLIB' ;              { name of file containing model library }

  LogOfe              = 0.4342944 ;                                { common logarithm of e }

  Esc                 = #27 ;                                          { code of Esc key }

  LF                  = ^M^J ;                                              { linefeed }

Type
  ParameterName       = String [ParameterNameLen] ;           { string for parameter name }
  ParameterRange      = 1 .. MaxParameter ;
  ParameterRangeW     = 0 .. MaxParameter ;
  TypeOfParameter     = (Endo, Exo, Cnst)  ;
  ParameterInfo       = Record
                          Name   : ParameterName   ;
                          Type_  : TypeOfParameter
                        End ;
  ParameterType       = Record                      { record holding parameters in the model }
                          ParNo     : ParameterRange ;
                          ParInfo   : Array [ParameterRange] Of ParameterInfo
                        End ;

  EntryType           = (Operator_, Parameter_, Number_) ;
  OperatorType        = (Plus_, BMinus_, UMinus_, Mult_, Div_, Power_, Sin_, Cos_, ArcTan_, Exp_,
                         Ln_, Log_, Sqr_, Sqrt_) ;
  PostfixEntryRange   = 1 .. MaxPostfixEntry ;
  PostfixEntryRangeW  = 0 .. MaxPostfixEntry ;
  PostfixEntry        = Record
                          Case Type_ : EntryType Of
                            Operator_  : (Operator : OperatorType) ;
                            Parameter_ : (Index    : ParameterRange) ;  { which parameter in array ParInfo }
                            Number_    : (Number : Real)
                        End ;
  PostfixType         = Record                            { record holding information about an }
                          EntryNo : PostfixEntryRangeW ;    { expression in postfix form          }
                          Entry   : Array [PostfixEntryRange] Of PostfixEntry
                        End ;

  SignTypeW           = (_neg, _zero, _pos, _undef) ;
  SignType            = (neg_, zero_, pos_) ;

  IMatrixRowType      = Array [ParameterRange] Of PostfixType ;     { one row of information matrix }

  InfixType           = String [255] ;                             { expression in infix form -  }
                                                                   { just an array of characters }

  EquationRange       = 1 .. MaxEquation ;
  EquationRangeW      = 0 .. MaxEquation ;
  EquationInfo        = Record
                          Expression : PostfixType    ;
                          Infix      : InfixType      ;
                          IMatrixRow : IMatrixRowType
                        End ;
  EquationType        = Record                          { record holding equations in the model }
                          EqnNo   : EquationRange ;
                          EqnInfo : Array [EquationRange] Of EquationInfo
                        End ;

  ModelNameType       = String [ModelNameLen] ;

  ModelType           = Record                         { record holding information about a model }
                          Name       : ModelNameType ;
                          Equations  : EquationType  ;
                          Parameters : ParameterType
                        End ;

  ModelTypePtr        = ^ ModelType ;

  ModelLibraryFileType = File Of ModelType ;

  IMatrixSType        = Array [EquationRange, ParameterRange] Of SignType ;
                                                            { I matrix in sign matrix form }

  EquationVectorType  = Array [EquationRange] Of Real ;                          { vector type }

  EquationMatrixType  = Array [EquationRange] Of EquationVectorType ;            { matrix type }

  ExoEndoMatrixType   = Array [ParameterRange] Of EquationVectorType ;        { matrix of gains }

  CrModelInfo         = Record                        { additional info about the current model }
                          IMatrixS      : IMatrixSType     ;
                          ExoEndoMatrix : ExoEndoMatrixType ;
                          ExoParNo ,                                 { # of exo vars/constants }
                          EndoParNo     : ParameterRangeW   ;        { # of endo vars }
                          ExoParAr ,                          { indirect index into array ParInfo for }
                          EndoParAr     : Array [ParameterRange] Of ParameterRange ;     { exos and endos }
                          Value         : Array [ParameterRange] Of Real               { system state }
                        End ;

  CrModelType         = Record
                          Model     : ModelType   ;
                          ModelInfo : CrModelInfo
                        End ;

  SignArrayType       = Array [ParameterRange] Of SignTypeW ;

  StringType          = String [255] ;

  SetOfChar           = Set Of Char  ;


Var
  ModelLibFile    : ModelLibraryFileType ;                              { file of model library }
  CrModel         : CrModelType          ;                                     { current model }

  CrModelGiven ,                                                        { current model loaded ? }
  IMatrixSGiven,                                                { I matrix reduced to sign matrix ? }
  ParValuesGiven  : Boolean ;                                        { system state specified ? }

  SPReg ,                         { integer variable holding stack pointer (SP),          }
                                  { used when error occurs to directly jump to main menu }
  HeapPtrSeg ,                    { segment & offset of heap pointer, used to allocate space on heap in }
  HeapPtrOfs      : Integer ;     { specific locations. Because when error occurs,we return to main menu}
                                  { without disallocating heap, so New causes heap to grow infinitely   }

{ *** returns upper case of string "s" *** }
Function UpCaseStr (s : StringType) : StringType ;
```

```
Var
  i : Byte ;
Begin
  For i:=1 To Length (s) Do
    s [i] := UpCase (s [i]) ;
  UpCaseStr := s
End ;

{ *** removes leading blanks in string "s" *** }
Function TrimL (s : StringType) : StringType ;
Begin
  While (s <> '') And (s [1] = ' ') Do
    Delete (s, 1, 1) ;
  TrimL := s
End ;

{ *** removes trailing blanks in string "s" *** }
Function TrimT (s : StringType) : StringType ;
Begin
  While (s <> '') And (s [Length (s)] = ' ') Do
    Delete (s, Length (s), 1) ;
  TrimT := s
End ;

{ *** returns a string of length Len formed from character Ch *** }
Function Spc (Ch  : Char ;
              Len : Byte  ) : StringType ;
Var
  s : StringType ;
  i : Byte       ;
Begin
  s := '' ;
  For i:=1 To Len Do
    s := s + Ch ;
  Spc := s
End ;

{ *** outputs the string s and controls if Ctrl-S is pressed. *** }
{ *** If so, waits until a key is pressed                     *** }
Procedure WriteSc (s : StringType) ;
Const
  CtrlS = #19 ;
Var
  Ch : Char ;
Begin
  Writeln (s) ;
  If KeyPressed Then
  Begin
    Read (Kbd, Ch) ;
    If Ch = CtrlS Then
      Read (Kbd, Ch)
  End
End ;

{ *** displays the error message corresponding to ErrorCode, waits until *** }
{ *** Esc is preessed, and returns to main menu directly. Number         *** }
{ *** additional information for some errors                             *** }
Procedure GiveError (ErrorCode,
                     Number    : Byte) ;
Var
  Ch : Char ;
Begin
  Window (1, 1, 80, 25) ;

  GotoXY (1, 25) ;
  Write  ('Error ', ErrorCode, ' : ') ;

  Case ErrorCode Of
    1  : Write ('unexpected end of equation') ;
```

```
      2  : Write ('unexpected character in equation') ;
      3  : Write ('improper equation syntax') ;
      4  : Write ('unmatched )') ;
      5  : Write (') expected') ;
      6  : Write ('sign of I matrix entry cannot be zero') ;
      7  : Write ('# of equations must be equal to # of endogenous vars') ;
      8  : Write ('cannot execute selection before a model is given') ;
      9  : Write ('cannot execute selection before sign of I matrix given') ;
      10 : Write ('cannot execute selection before par values are given') ;
      11 : Write ('equations do not converge for the given initial values') ;
      12 : Write ('singular matrix encountered in analysis') ;
      13 : Write ('error in division operator') ;
      14 : Write ('error in power operator') ;
      15 : Write ('error in ln operator') ;
      16 : Write ('error in log operator') ;
      17 : Write ('error in sqrt operator') ;
      18 : Write ('model does not exist') ;
      19 : Write ('model already exists') ;
      20 : Write ('invalid selection') ;
      21 : Write ('number of equations cannot be greater than ', MaxEquation) ;
      22 : Write ('number of parameters cannot be greater than ', MaxParameter) ;
      23 : Write ('number of parameters is zero') ;
      24 : Write ('no such parameter in the current model') ;
      25 : Write ('parameter must be an exogenous var or a constant') ;
      26 : Write ('# of unknowns must be equal to # of equations') ;
      50 : Write ('user break, command aborted')
    End ;

    Case ErrorCode Of
      1 .. 5 : Write (' (Position=', Number, ')') ;
      6      : With CrModel. Model. Parameters Do
                 Write (' (Eqn:', (Number-1) Div ParNo + 1, ',Par:',
                        TrimT (ParInfo [(Number-1) Mod ParNo + 1]. Name), ')')
    End ;

    GotoXY (68, 25) ;
    Write ('Press Esc...') ;

    Repeat
      Read (Kbd, Ch)
    Until (Ch = Esc) And (Not KeyPressed) ;

    Inline ($89/$EC/            { L: MOV SP,BP    }    { SP is saved into BP when a procedure is called }
            $5D/                {    POP BP       }    { BP is saved into stack }
            $39/$26/SPReg/      {    CMP SP,SPReg }    { SPReg contains address of top of stack }
            $75/$F7/            {    JNZ L        }    { if not stack emptied yet, jump to L }
            $C3         )       {    RET          }    { now stack holds address of main menu, }
  End ;                                                { return to main menu }

  { *** displays a message corresponding to MsgCode indicating the *** }
  { *** successful completion of a main menu operation             *** }
  Procedure Message (MsgCode : Byte) ;
  Var
    Ch : Char ;
  Begin
    Window (1, 1, 80, 25) ;

    GotoXY (1, 25) ;

    Case MsgCode Of
      1  : Write ('signs of I matrix entries are initialized') ;
      2  : Write ('parameter values are initialized') ;
      3  : Write ('sign analysis of total differentials completed') ;
      4  : Write ('current model is inserted to the library') ;
      5  : Write ('listing completed') ;
      6  : Write ('model is deleted from library') ;
      7  : Write ('model is inserted to the library') ;
      8  : Write ('model from library is loaded as the current model') ;
      9  : Write ('model given by user is loaded as the current model') ;
```

```
     10 : Write ('sign analysis of gains completed') ;
     11 : Write ('value analysis completed') ;
     12 : Write ('listing of gains completed') ;
     13 : Write ('listing of elasticities completed') ;
   End ;

   GotoXY (68, 25) ;
   Write ('Press Esc...') ;

   Repeat
     Read (Kbd, Ch)
   Until (Ch = Esc) And (Not KeyPressed)
 End ;

 { *** line editor, inputs a string of length Len position of cursor *** }
 Procedure ReadInput (    Len      : Byte       ;
                      Var InputStr : StringType ) ;
 Const
   Home = #71 ;   End_  = #79 ;   CR   = #13 ;   BackSp = #8 ;
   Left = #75 ;   Right = #77 ;   Del  = #83 ;   CtrlC  = #3 ;
 Var
   Row, Col,
   x  , y   : Byte    ;
   EndInput : Boolean ;
   Ch       : Char    ;

   Procedure DeleteChar (CharNo : Byte) ;
   Begin
     InputStr := Copy (InputStr,1,CharNo-1) + Copy (InputStr,CharNo+1,Len-CharNo) + ' ' ;
     GotoXY (Col, Row) ;
     Write  (InputStr)
   End ;

 Begin
   InputStr := Spc (' ', Len) ;

   Row := WhereY ;
   Col := WhereX ;
   y   := Row    ;
   x   := Col    ;
   EndInput := False ;

   Repeat
     GotoXY (x, y) ;
     Read (Kbd,Ch) ;
     Case Ch of
       Esc : If KeyPressed Then
                Begin
                  Read (Kbd,Ch) ;
                  Case Ch of
                    Left     : If x <> Col Then
                                  x := x - 1 ;
                    Right    : If x <> Col+Len-1 Then
                                  x := x + 1 ;
                    End_     : x := Col + Len - 1 ;
                    Home     : x := Col ;
                    Del      : DeleteChar (x-Col+1)
                  End { Case Ch of }
                End ; { If ... Then }
       BackSp : If x <> Col Then
                  Begin
                    DeleteChar (x-Col) ;
                    x := x - 1
                  End ;
       CR     : EndInput := True ;
       CtrlC  : GiveError (50, 0) ;                       { command aborted by user }
       Else     Begin
                  Ch := UpCase (Ch) ;
                  Write (Ch) ;
```

```
               InputStr [x-Col+1] := Ch ;
               If x <> Col+Len-1 Then
                 x := x + 1
             End
    End { Case }
  Until EndInput ;

  Writeln
End ;

{ *** reads an integer, returns in Number *** }
Procedure ReadInputI (    Len    : Byte    ;
                      Var Number : Integer  ) ;
Var
  InputStr : StringType ;
  Result   : Integer    ;
Begin
  ReadInput (Len, InputStr) ;
  Val (InputStr, Number, Result)
End ;

{ *** reads a real, returns in Number *** }
Procedure ReadInputR (    Len    : Byte ;
                      Var Number : Real  ) ;
Var
  InputStr : StringType ;
  Result   : Integer    ;
Begin
  ReadInput (Len, InputStr) ;
  Val (InputStr, Number, Result)
End ;

{ *** reads a string, returns in InputStr *** }
Procedure ReadInputS (    Len      : Byte       ;
                      Var InputStr : StringType  ) ;
Begin
  ReadInput (Len, InputStr)
End ;

{ *** reads a character, returns in Ch. ChSet contains legal characters *** }
Procedure ReadChar (Var Ch    : Char       ;
                        ChSet : SetOfChar   ) ;
Var
  InputStr : StringType ;
  Row, Col : Byte       ;
Begin
  Row := WhereY ;
  Col := WhereX ;
  Repeat
    GotoXY (Col, Row) ;
    ReadInput (1, InputStr)
  Until (InputStr [1] IN ChSet) ;
  Ch := InputStr [1]
End ;

{ *** Is Operator a binary or unary operator *** }
Function BinaryOperator (Operator : OperatorType) : Boolean ;
Begin
  BinaryOperator := (Operator IN [Plus_, BMinus_, Mult_, Div_, Power_])
End ;

{ *** Equation NoEqn of model Model contains the parameter whose index is NoPar ? *** }
{ *** It contains if the parameter appears in the infix form of the equation     *** }
Function ParInEqn (Var Model : ModelType      ;
                       NoPar : ParameterRange ;
                       NoEqn : EquationRange   ) : Boolean ;
Begin
  ParInEqn := (Pos (TrimT (Model. Parameters. ParInfo [NoPar]. Name),
               Model. Equations. EqnInfo [NoEqn]. Infix) <> 0)
```

```
End ;

{ *** The following five functions are for operators that may result in an error *** }

{ *** division operator *** }
Function DivOp (Operand1, Operand2 : Real) : Real ;
Begin
  If Operand2 = 0
    Then GiveError (13, 0)
    Else DivOp := Operand1 / Operand2
End ;

{ *** power operator *** }
Function PowerOp (Operand1, Operand2 : Real) : Real ;
Var
  Result : Real ;
Begin
  If (Operand1 > 0) Or ((Operand1 <0) And (Operand2 = Int (Operand2))) Then
  Begin
    Result := Exp (Operand2 * Ln (Abs (Operand1))) ;
    If (Operand1 < 0) And (Odd (Round (Abs (Operand2)))) Then
      Result := - Result
  End
  Else
    If (Operand1 = 0) And (Operand2 > 0)
      Then Result := 0
      Else GiveError (14, 0)
End ;

{ *** ln operator *** }
Function LnOp (Operand : Real) : Real ;
Begin
  If Operand <= 0
    Then GiveError (15, 0)
    Else LnOp := Ln (Operand)
End ;

{ *** log operator *** }
Function LogOp (Operand : Real) : Real ;
Begin
  If Operand <= 0
    Then GiveError (16, 0)
    Else LogOp := Ln (Operand) * LogOfe
End ;

{ *** square root operator *** }
Function SqrtOp (Operand : Real) : Real ;
Begin
  If Operand < 0
    Then GiveError (17, 0)
    Else SqrtOp := Sqrt (Operand)
End ;

{ *** returns the result of an operator *** }
Function OpResult (Operand1 ,
                   Operand2  : Real         ;
                   Operator  : OperatorType  ) : Real ;
Begin
  Case Operator Of
    UMinus_ : OpResult := - Operand2          ;
    Sin_    : OpResult := Sin    (Operand2) ;
    Cos_    : OpResult := Cos    (Operand2) ;
    ArcTan_ : OpResult := ArcTan (Operand2) ;
    Exp_    : OpResult := Exp    (Operand2) ;
    Ln_     : OpResult := LnOp   (Operand2) ;
    Log_    : OpResult := LogOp  (Operand2) ;
    Sqr_    : OpResult := Sqr    (Operand2) ;
    Sqrt_   : OpResult := SqrtOp (Operand2) ;
    Plus_   : OpResult := Operand1 + Operand2 ;
    BMinus_ : OpResult := Operand1 - Operand2 ;
    Mult_   : OpResult := Operand1 * Operand2 ;
    Div_    : OpResult := DivOp   (Operand1, Operand2) ;
    Power_  : OpResult := PowerOp (Operand1, Operand2)
  End
End ;

{ *** returns the inverse of sign Sign *** }
Function SignInverse (Sign : SignType) : SignType ;
Begin
  Case Sign Of
    neg_  : SignInverse := pos_  ;
    zero_ : SignInverse := zero_ ;
    pos_  : SignInverse := neg_
  End
End ;

{ *** Library contains a model whose name is Name ? If so, return *** }
{ *** the file index in Index, else Index is undetermined.        *** }
Function ModelExists (    Name  : ModelNameType ;
                      Var Index : Integer       ) : Boolean ;
Var
  Model : ModelTypePtr ;
  Found : Boolean      ;
Begin
  Found := False ;
  Reset (ModelLibFile) ;

  Model := Ptr (HeapPtrSeg, HeapPtrOfs + $7FFF) ; { create space on heap for the model. Space is 32K      }
                                                  { beyond the beginning of heap, since top of heap may   }
                                                  { contains another model before this function is called }
  While (Not Found) And (Not Eof (ModelLibFile)) Do
  Begin
    Read (ModelLibFile, Model^) ;
    If Model^. Name = Name Then
    Begin
      Found := True ;
      Index := FilePos (ModelLibFile) - 1
    End
  End ;

  ModelExists := Found
End ;

{ parameter whose name is Name exists in Parameters ?     *** }
{ If so, return its index in Index, else return ParNo + 1 *** }
Function ParameterExists (    Parameters : ParameterType  ;
                              Name       : ParameterName  ;
                          Var Index      : ParameterRange ) : Boolean ;
Var
  Found : Boolean ;
Begin
  With Parameters Do
  Begin
    Found := False ;
    Index := 1 ;

    While (Not Found) And (Index <= ParNo) Do
      If ParInfo [Index]. Name = Name
        Then Found := True
        Else Index := Index + 1 ;

    ParameterExists := Found
  End
End ;

{ *** postfix expression Item is formed from a single number *** }
Procedure NumberEntry (Var Item : PostfixType ;
                           Num  : Real        ) ;
```

```
Begin
  With Item, Entry [1] Do
  Begin
    EntryNo := 1 ;
    Type_   := Number_ ;
    Number  := Num
  End
End ;

{ *** postfix expression Item is formed from a single operator *** }
Procedure OperatorEntry (Var Item : PostfixType  ;
                             Ope  : OperatorType  ) ;
Begin
  With Item, Entry [1] Do
  Begin
    EntryNo  := 1 ;
    Type_    := Operator_ ;
    Operator := Ope
  End
End ;

{ *** solves Ax=b for x by Gauss Elimination. No is # of unknowns *** }
Procedure GaussElimination (    A  : EquationMatrixType ;
                                b  : EquationVectorType ;
                                No : EquationRange      ;
                            Var x  : EquationVectorType  ) ;
Var
  i, j, k    : EquationRange ;
  MaxRowNo   : EquationRange ;
  Multiplier : Real          ;

  Procedure InterchangeRows (RowNo1 ,
                             RowNo2  : EquationRange) ;
  Var
    Vector : EquationVectorType ;
    Number : Real               ;
  Begin
    Vector     := A [RowNo1] ;
    A [RowNo1] := A [RowNo2] ;
    A [RowNo2] := Vector     ;

    Number     := b [RowNo1] ;
    b [RowNo1] := b [RowNo2] ;
    b [RowNo2] := Number
  End ;

Begin
  For i:=1 To No-1 Do
  Begin
    MaxRowNo := i ;

    For j:=i+1 To No Do
      If Abs (A [j,i]) > Abs (A [MaxRowNo,i]) Then
        MaxRowNo := j ;

    If MaxRowNo <> i Then
      InterchangeRows (MaxRowNo, i) ;

    If A [i,i] = 0 Then
      GiveError (12, 0) ;

    For j:=i+1 To No Do
    Begin
      Multiplier := A [j,i] / A [i,i] ;

      For k:=i+1 To No Do
        A [j,k] := A [j,k] - Multiplier * A [i,k] ;

      b [j] := b [j] - Multiplier * b [i]
    End
  End ;

  For i:=No DownTo 1 Do
  Begin
    x [i] := b [i] ;

    For j:=i+1 To No Do
      x [i] := x [i] - A [i,j] * x [j] ;

    If A [i,i] = 0 Then
      GiveError (12, 0) ;

    x [i] := x [i] / A [i,i]
  End
End ;

{ *** multiplies matrix A by vector x, returns result in b *** }
Procedure MatrixVectorMult (     No : EquationRange ;
                                 A  : EquationMatrixType ;
                                 x  : EquationVectorType ;
                             Var b  : EquationVectorType   ) ;
Var
  i, j : EquationRange ;
Begin
  For i:=1 To No Do
  Begin
    b [i] := 0 ;
    For j:=1 To No Do
      b [i] := b [i] + A [i,j] * x [j]
  End
End ;

{ *** computes the value of the postfix expression. For parameters, system state is used *** }
Function EvaluateExpression (Expression : PostfixType) : Real ;
Const
  MaxStackItem = 50 ;
Type
  StackItemRange  = 1 .. MaxStackItem ;
  StackItemRangeW = 0 .. MaxStackItem ;
  StackType       = Record
                      Top  : StackItemRangeW ;
                      Item : Array [StackItemRange] Of Real
                    End ;
Var
  Stack      : StackType          ;
  NoEntry    : PostfixEntryRange  ;
  Operand1 ,
  Operand2   : Real               ;

  Procedure Push (Var Stack  : StackType ;
                      Number : Real       ) ;
  Begin
    With Stack Do
    Begin
      Top := Top + 1 ;
      Item [Top] := Number
    End
  End ;

  Function Pop (Var Stack : StackType) : Real ;
  Begin
    With Stack Do
    Begin
      Pop := Item [Top] ;
      Top := Top - 1
    End
  End ;
```

```
Begin  { EvaluateExpression }
  With Stack, Expression Do
  Begin
    Top := 0 ;

    For NoEntry:=1 To EntryNo Do
      With Entry [NoEntry] Do
        If Type_ = Parameter_ Then
          Push (Stack, CrModel. ModelInfo. Value [Index])
        Else
          If Type_ = Number_ Then
            Push (Stack, Number)
          Else Begin
            Operand2 := Pop (Stack) ;
            If BinaryOperator (Operator) Then
              Operand1 := Pop (Stack) ;

            Push (Stack, OpResult (Operand1, Operand2, Operator))
          End ;

    EvaluateExpression := Pop (Stack)

  End { With }
End ;

{ *** parses equation string, inserts parameters into the model *** }
Procedure ParseEquation (Var Model  : ModelType     ;
                             NoEqn  : EquationRange ;
                             EqnStr : StringType     ) ;
Const
  MaxStackItem = 50 ;
Type
  TokenType       = (_Plus, _BMinus, _UMinus, _Mult, _Div, _Power,       { first 14 entries of TokenType }
                     _Sin, _Cos, _ArcTan, _Exp, _Ln, _Log, _Sqr, _Sqrt,   { must be in the same order as  }
                     _LeftPr, _RightPr, _Parameter, _Number) ;            { in OperatorType               }
  StackItemRange  = 1 .. MaxStackItem ;
  StackItemRangeW = 0 .. MaxStackItem ;
  StackType       = Record
                      Top  : StackItemRangeW ;
                      Item : Array [StackItemRange] Of TokenType
                    End ;
  StateType       = 0 .. 3 ;
Var
  Token       : TokenType     ;
  Stack       : StackType     ;
  TokenSymbol : ParameterName ;
  State       : StateType     ;
  PrCount     : Integer       ;
  EqnPos      : Byte          ;

  Procedure DelStr (Var s    : StringType ;
                        Pos1,
                        Pos2 : Byte       ) ;
  Begin
    Delete (s, Pos1, Pos2) ;
    EqnPos := EqnPos + (Pos2 - Pos1 + 1)
  End ;

  Function TrimL (s : StringType) : StringType ;
  Begin
    While (s <> '') And (s [1] = ' ') Do
      DelStr (s, 1, 1) ;
    TrimL := s
  End ;

  Function GetToken : TokenType ;
  Var
    TokenLen   : Integer ;
  Begin
  EqnStr := TrimL (EqnStr) ;

  TokenLen := 1 ;

  If EqnStr = '' Then
    GiveError (1, EqnPos) ;

  Case EqnStr [1] Of
    'A'..'Z' : Begin
                 While (TokenLen < Length (EqnStr)) And (EqnStr [TokenLen+1] IN ['A'..'Z','0'..'9']) Do
                    TokenLen := TokenLen + 1 ;
                 TokenSymbol := Copy (EqnStr, 1, TokenLen) ;
                 If TokenSymbol = 'SIN'    Then GetToken := _Sin    Else
                 If TokenSymbol = 'COS'    Then GetToken := _Cos    Else
                 If TokenSymbol = 'ARCTAN' Then GetToken := _ArcTan Else
                 If TokenSymbol = 'EXP'    Then GetToken := _Exp    Else
                 If TokenSymbol = 'LN'     Then GetToken := _Ln     Else
                 If TokenSymbol = 'LOG'    Then GetToken := _Log    Else
                 If TokenSymbol = 'SQR'    Then GetToken := _Sqr    Else
                 If TokenSymbol = 'SQRT'   Then GetToken := _Sqrt   Else
                   GetToken := _Parameter
               End ;
    '0'..'9' : Begin
                 While (TokenLen < Length (EqnStr)) And (EqnStr [TokenLen+1] IN ['0'..'9']) Do
                   TokenLen := TokenLen + 1 ;
                 If EqnStr [TokenLen+1] = '.' Then
                 Begin
                   TokenLen := TokenLen + 1 ;
                   While (TokenLen < Length (EqnStr)) And (EqnStr [TokenLen+1] IN ['0'..'9']) Do
                     TokenLen := TokenLen + 1
                 End ;
                 If EqnStr [TokenLen+1] = 'E' Then
                 Begin
                   TokenLen := TokenLen + 1 ;
                   If EqnStr [TokenLen+1] IN ['-','+'] Then
                     TokenLen := TokenLen + 1 ;
                   While (TokenLen < Length (EqnStr)) And (EqnStr [TokenLen+1] IN ['0'..'9']) Do
                     TokenLen := TokenLen + 1
                 End ;
                 GetToken := _Number
               End ;
    '+'      : GetToken := _Plus    ;
    '-'      : If State = 0
                 Then GetToken := _UMinus
                 Else GetToken := _BMinus   ;
    '*'      : GetToken := _Mult    ;
    '/'      : GetToken := _Div     ;
    '^'      : GetToken := _Power   ;
    '('      : GetToken := _LeftPr  ;
    ')'      : GetToken := _RightPr ;
   Else        GiveError (2, EqnPos)
  End ; { Case }

  TokenSymbol := Copy (EqnStr, 1, TokenLen) ;

  DelStr (EqnStr, 1, TokenLen)
End ;

Function StackEmpty (Var Stack : StackType) : Boolean ;
Begin
  StackEmpty := (Stack. Top = 0)
End ;

Procedure Push (Var Stack    : StackType ;
                    Operator : TokenType  ) ;
Begin
  With Stack Do
  Begin
    Top := Top + 1 ;
```

```
      Item [Top] := Operator
    End
  End ;

  Function Pop (Var Stack  : StackType) : TokenType ;
  Begin
    With Stack Do
    Begin
      Pop := Item [Top] ;
      Top := Top - 1
    End
  End ;

  { *** precedence of operators *** }
  Function Prcd (Operator1, Operator2 : TokenType) : Boolean ;
  Begin
    Case Operator1 Of
      _Sin   , _Cos ,
      _ArcTan, _Exp ,
      _Ln    , _Log ,
      _Sqr   , _Sqrt : Prcd := False ;      { Operator2 may only be ( }
      _LeftPr        : Prcd := False ;
      _UMinus        : Prcd := (Operator2 IN [_RightPr, _Plus, _BMinus, _Mult, _Div, _Power, _UMinus]) ;
      _Power         : Prcd := (Operator2 IN [_RightPr, _Plus, _BMinus, _Mult, _Div, _Power]) ;
      _Mult, _Div    : Prcd := (Operator2 IN [_RightPr, _Plus, _BMinus, _Mult, _Div]) ;
      _Plus, _BMinus : Prcd := (Operator2 IN [_RightPr, _Plus, _BMinus])
    End
  End ;

  Procedure AddToPostfixExpr (TypeOfEntry    : EntryType     ;
                                TokenStr       : ParameterName ;
                              TypeOfOperator : OperatorType  ) ;
  Var
    ResultCode : Integer ;

    Procedure ProcessParameter (    ParName : ParameterName ;
                                Var Index   : ParameterRange ) ;
    Var
      Found : Boolean        ;
      No    : ParameterRange ;
    Begin
      ParName := ParName + Spc (' ', ParameterNameLen) ;

      With Model, Parameters Do
        If Not ParameterExists (Parameters, ParName, Index) Then  { assigns index }
        Begin
          If ParNo = MaxParameter Then
            GiveError (22, 0) ;
          ParNo := ParNo + 1 ;
          ParInfo [ParNo]. Name := ParName
        End
    End ;

  Begin  { AddToPostfixExpr }
    With Model. Equations. EqnInfo [NoEqn]. Expression Do
    Begin
      EntryNo := EntryNo + 1 ;

      With Entry [EntryNo] Do
      Begin
        Type_ := TypeOfEntry ;
        Case Type_ Of
          Operator_  : Operator := TypeOfOperator ;
          Parameter_ : ProcessParameter (TokenStr, Index) ;
          Number_    : Val (TokenStr, Number, ResultCode)
        End
      End
    End
  End ;
```

```
Begin  { ParseEquation }
  EqnPos := 1 ;

  Model. Equations. EqnInfo [NoEqn]. Expression. EntryNo := 0 ;

  With Stack Do
  Begin
    Top := 0 ;

    State   := 0 ;
    PrCount := 0 ;

    Repeat
      Token := GetToken ;


      {                                                                           }
      {  S0  -->  ( S0  ¦  Parameter S2  ¦  Number S2  ¦  Sin .. Sqrt S3  ¦  - S1 }
      {  S1  -->  ( S0  ¦  Parameter S2  ¦  Number S2  ¦  Sin .. Sqrt S3          }
      {  S2  -->  ) S2  ¦  +-*/^ S1                                               }
      {  S3  -->  ( S0                                                            }
      {                                                                           }


      If ((State=0) And (Not (Token IN [_LeftPr,_Parameter,_Number,_Sin,_Cos,_ArcTan,_Exp,_Ln,_Log,_Sqr,
                                                                                  _Sqrt,_UMinus]))) Or
         ((State=1) And (Not (Token IN [_LeftPr,_Parameter,_Number,_Sin,_Cos,_ArcTan,_Exp,_Ln,_Log,_Sqr,
                                                                                          _Sqrt]))) Or
         ((State=2) And (Not (Token IN [_RightPr,_Plus,_BMinus,_Mult,_Div,_Power]))) Or
         ((State=3) And (Token <> _LeftPr)) Then
        GiveError (3, EqnPos) ;

      Case State Of
        0 : Case Token Of
              _LeftPr     : PrCount := PrCount + 1 ;
              _Parameter,
              _Number     : State := 2 ;
              _UMinus     : State := 1 ;
             Else           State := 3
            End ;
        1 : Case Token Of
              _LeftPr     : Begin
                              PrCount := PrCount + 1 ;
                              State   := 0
                            End ;
              _Parameter,
              _Number     : State := 2 ;
             Else           State := 3
            End ;
        2 : Case Token Of
              _RightPr : PrCount := PrCount - 1 ;
             Else        State := 1
            End ;
        3 : Begin
              PrCount := PrCount + 1 ;
              State := 0
            End
      End ; { Case }

      If Token = _Parameter Then
        AddToPostfixExpr (Parameter_, TokenSymbol, Plus_)   { _Plus dummy }
      Else
        If Token = _Number Then
          AddToPostfixExpr (Number_, TokenSymbol, Plus_)    { _Plus dummy }
        Else Begin  { operator }
          While (Not StackEmpty (Stack)) And (Prcd (Item [Top], Token)) Do
            AddToPostfixExpr (Operator_, '', OperatorType (Ord (Pop (Stack)))) ;
          If Token = _RightPr Then
          Begin
            If StackEmpty (Stack) Then
              GiveError (4, EqnPos) ;
            Top := Top - 1 ;   { delete matching ( from stack }
            If (Not StackEmpty(Stack)) And (Item[Top] IN [_Sin,_Cos,_ArcTan,_Exp,_Ln,_Log,_Sqr,_Sqrt]) Then
              AddToPostfixExpr (Operator_, '', OperatorType (Ord (Pop (Stack))))
          End
          Else
            Push (Stack, Token)
        End
    Until (TrimL (EqnStr) = '') ;

    If State <> 2 Then
      GiveError (1, EqnPos) ;

    If PrCount <> 0 Then
      GiveError (5, EqnPos) ;

    While Not (StackEmpty (Stack)) Do
      AddToPostfixExpr (Operator_, '', OperatorType (Ord (Pop (Stack))))
  End { With }
End ;

{ *** converts I matrix into a sign matrix by restricting system space *** }
{ *** signs of expressions are asked for when needed                   *** }
Procedure SignOfInformationMatrix ;
Const
  MaxExpr = 50 ;
Type
  ExprNoRange  = 1 .. MaxExpr ;
  ExprNoRangeW = 0 .. MaxExpr ;
  ExprSignType = Record                                { holds expressions whose signs are asked }
                   ExprNo : ExprNoRangeW ;
                   ExprAr : Array [ExprNoRange] Of Record
                                                     Expr : InfixType ;
                                                     Sign : SignType
                                                   End
                 End ;
Var
  ExprSign      : ExprSignType   ;
  NoEqn         : EquationRange  ;
  NoPar         : ParameterRange ;

  Procedure SignOfExpression (    Expression : PostfixType ;
                              Var Sign       : SignType    ) ;
  Var
    Expr1, Expr2  : PostfixType        ;
    Sign1, Sign2  : SignType           ;
    NoEntry       : PostfixEntryRangeW ;
    Counter       : Integer            ;

    Procedure ShiftEntries (    Expr1   : PostfixType        ;
                                Index1 ,
                                Index2  : PostfixEntryRangeW ;
                            Var Expr2   : PostfixType        ) ;
    Var
      i : PostfixEntryRangeW ;
    Begin
      Expr2. EntryNo := Index2-Index1+1 ;

      For i:=1 To Expr2. EntryNo Do
        Expr2. Entry [i] := Expr1. Entry [Index1+i-1]
    End ;

    Function NumberSign (Number : Real) : SignType ;
    Begin
      If Number < 0 Then
        NumberSign := neg_
  Else
    If Number = 0
      Then NumberSign := zero_
      Else NumberSign := pos_
End ;

Function SignNumber (Sign : SignType) : Integer ;
Begin
  Case Sign Of
    neg_  : SignNumber := -1 ;
    zero_ : SignNumber :=  0 ;
    pos_  : SignNumber :=  1
  End
End ;

Function ReadSign (Expression : PostfixType) : SignType ;
Var
  InfixExpr : InfixType   ;
  No        : ExprNoRange ;
  Found     : Boolean     ;
  SignCh    : Char        ;

  Function ConvertToInfix (Expression : PostfixType) : InfixType ;
  Const
    MaxStackItem = 50 ;
  Type
    StackItemRange  = 1 .. MaxStackItem ;
    StackItemRangeW = 0 .. MaxStackItem ;
    StackType       = Record
                        Top  : StackItemRangeW ;
                        Item : Array [StackItemRange] Of StringType
                      End ;
  Var
    Stack      : StackType         ;
    NoEntry    : PostfixEntryRange ;
    NumberStr ,
    Str1, Str2 : StringType        ;

    Procedure Push (Var Stack : StackType  ;
                        Str   : StringType  ) ;
    Begin
      With Stack Do
      Begin
        Top := Top + 1 ;
        Item [Top] := Str
      End
    End ;

    Function Pop (Var Stack : StackType) : StringType ;
    Begin
      With Stack Do
      Begin
        Pop := Item [Top] ;
        Top := Top - 1
      End
    End ;

    Function OperatorStr (Operator : OperatorType) : StringType ;
    Begin
      Case Operator Of
        Plus_   : OperatorStr := '+'      ;
        BMinus_,
        UMinus_ : OperatorStr := '-'      ;
        Mult_   : OperatorStr := '*'      ;
        Div_    : OperatorStr := '/'      ;
        Power_  : OperatorStr := '^'      ;
        Sin_    : OperatorStr := 'Sin'    ;
        Cos_    : OperatorStr := 'Cos'    ;
        ArcTan_ : OperatorStr := 'ArcTan' ;
```

```
          Exp_    : OperatorStr := 'Exp'    ;
          Ln_     : OperatorStr := 'Ln'     ;
          Log_    : OperatorStr := 'Log'    ;
          Sqr_    : OperatorStr := 'Sqr'    ;
          Sqrt_   : OperatorStr := 'Sqrt'
        End
      End ;

    Begin  { ConvertToInfix }
      With Stack, Expression Do
      Begin
        Top := 0 ;

        For NoEntry:=1 To EntryNo Do
          With Entry [NoEntry] Do
            Case Type_ Of
              Parameter_ : Push (Stack, TrimT (CrModel. Model. Parameters. ParInfo [Index]. Name)) ;
              Number_    : Begin
                             Str  (Number :1:2, NumberStr) ;
                             Push (Stack, NumberStr)
                           End ;
              Operator_  : Begin
                             Str2 := Pop (Stack) ;
                             If BinaryOperator (Operator)
                               Then Str1 := Pop (Stack)
                               Else Str1 := '' ;
                             If Operator IN [Sin_, Cos_, ArcTan_, Exp_, Ln_, Log_, Sqr_, Sqrt_] Then
                               If Str2 [1] = '('
                                 Then Push (Stack, OperatorStr (Operator) + Str2)
                                 Else Push (Stack, OperatorStr (Operator) + '(' + Str2 + ')')
                             Else
                               Push (Stack, '(' + Str1 + OperatorStr (Operator) + Str2 + ')')
                           End
            End ; { Case }

        ConvertToInfix := Item [1]

      End { With }
    End ;

  Begin  { ReadSign }
    With ExprSign Do
    Begin
      InfixExpr := ConvertToInfix (Expression) ;

      Found := False ;
      No := 1 ;

      While (Not Found) And (No <= ExprNo) Do
        If ExprAr [No]. Expr = InfixExpr
          Then Found := True
          Else No := No + 1 ;

      If Found Then
        ReadSign := ExprAr [No]. Sign
      Else Begin
        Write ('Sign of ', InfixExpr, ' (- 0 +) : ') ;
        ReadChar (SignCh, ['-','0','+']) ;
        ExprNo := ExprNo + 1 ;
        With ExprAr [ExprNo] Do
        Begin
          Case SignCh Of
            '-' : Sign := neg_  ;
            '0' : Sign := zero_ ;
            '+' : Sign := pos_
          End ;
          Expr := InfixExpr ;
          ReadSign := Sign
        End { With }
        End { Else Begin }
      End { With }
    End ;


  Begin  { SignOfExpression }
    With Expression Do
      If EntryNo = 1 Then
        Case Entry [1]. Type_ Of
          Parameter_ : Sign := ReadSign (Expression) ;         { signs of parameters are given by user }
          Number_    : Sign := NumberSign (Entry [1]. Number)
        End
      Else Begin
        NoEntry := EntryNo - 1 ;
        Counter := 1 ;
        While (Counter > 0) Do                             { find the operator with lowest precedence }
        Begin
          If Entry [NoEntry]. Type_ <> Operator_ Then
            Counter := Counter - 1
          Else
            If BinaryOperator (Entry [NoEntry]. Operator) Then
              Counter := Counter + 1 ;
          NoEntry := NoEntry - 1
        End ;

        ShiftEntries (Expression, 1          , NoEntry    , Expr1) ;   { first & second parts. If a unary }
        ShiftEntries (Expression, NoEntry + 1, EntryNo - 1, Expr2) ;   { operator, then Expr1 is not used }

        Case Entry [EntryNo]. Operator Of
          Plus_     : Begin
                        SignOfExpression (Expr1, Sign1) ;
                        SignOfExpression (Expr2, Sign2) ;
                        If ((Sign1 = pos_) And (Sign2 = neg_)) Or ((Sign1 = neg_) And (Sign2 = pos_))
                          Then Sign := ReadSign (Expression)
                          Else Sign := NumberSign (SignNumber (Sign1) + SignNumber (Sign2))
                      End ;
          BMinus_   : Begin
                        SignOfExpression (Expr1, Sign1) ;
                        SignOfExpression (Expr2, Sign2) ;
                        If ((Sign1 = pos_) And (Sign2 = pos_)) Or ((Sign1 = neg_) And (Sign2 = neg_))
                          Then Sign := ReadSign (Expression)
                          Else Sign := NumberSign (SignNumber (Sign1) - SignNumber (Sign2))
                      End ;
          UMinus_   : Begin
                        SignOfExpression (Expr2, Sign2) ;
                        Sign := SignInverse (Sign2)
                      End ;
          Mult_     : Begin
                        SignOfExpression (Expr1, Sign1) ;
                        If Sign1 = zero_ Then
                          Sign := zero_
                        Else Begin
                          SignOfExpression (Expr2, Sign2) ;
                          Sign := NumberSign (SignNumber (Sign1) * SignNumber (Sign2))
                        End
                      End ;
          Div_      : Begin
                        SignOfExpression (Expr1, Sign1) ;
                        SignOfExpression (Expr2, Sign2) ;
                        Sign := NumberSign (DivOp (SignNumber (Sign1), SignNumber (Sign2)))
                      End ;
          Power_    : Begin
                        SignOfExpression (Expr1, Sign1) ;
                        If Sign1 = pos_ Then
                          Sign := pos_
                        Else Begin
                          SignOfExpression (Expr2, Sign2) ;
                          If (Sign1 = neg_) And (Sign2 IN [pos_, neg_])
                            Then Sign := ReadSign (Expression)
                        Else Sign := NumberSign (PowerOp (SignNumber (Sign1), SignNumber (Sign2)))
                    End
                  End ;
        Sin_      : Begin
                      SignOfExpression (Expr2, Sign2) ;
                      If Sign2 = zero_
                        Then Sign := zero_
                        Else Sign := ReadSign (Expression)
                    End ;
        Cos_      : Begin
                      SignOfExpression (Expr2, Sign2) ;
                      If Sign2 = zero_
                        Then Sign := pos_
                        Else Sign := ReadSign (Expression)
                    End ;
        Exp_      : Sign := pos_ ;
        ArcTan_   : SignOfExpression (Expr2, Sign) ;
        Sqr_      : Begin
                      SignOfExpression (Expr2, Sign2) ;
                      If Sign2 = zero_
                        Then Sign := zero_
                        Else Sign := pos_
                    End ;
        Sqrt_     : Begin
                      SignOfExpression (Expr2, Sign2) ;
                      Sign := NumberSign (SqrtOp (SignNumber (Sign2)))
                    End ;
        Ln_, Log_ : Begin
                      SignOfExpression (Expr2, Sign2) ;
                      If Sign2 = pos_
                        Then Sign := ReadSign (Expression)
                        Else Sign := NumberSign (OpResult(0, SignNumber(Sign2), Entry[EntryNo].Operator))
                    End
      End { Case }
    End { Else Begin }
  End ;

Begin  { SignOfInformationMatrix }
  ExprSign. ExprNo := 0 ;

  IMatrixSGiven := False ;

  With CrModel, Model, ModelInfo, Equations, Parameters Do
    For NoEqn:=1 To EqnNo Do
      For NoPar:=1 To ParNo Do
        If ParInEqn (Model, NoPar, NoEqn) Then
        Begin
          SignOfExpression (EqnInfo [NoEqn]. IMatrixRow [NoPar], IMatrixS [NoEqn, NoPar]) ;
          If IMatrixS [NoEqn, NoPar] = zero_ Then
            GiveError (6, (NoEqn-1) * ParNo + NoPar)
        End ;

  Message (1) ;

  IMatrixSGiven := True
End ;

{ *** sign analysis on the current model. In this procedure exo and endo means parameters *** }
{ *** whose signs are given and parameters whose signs are unknown, respectively         *** }
Procedure SignAnalysis (Signs : SignArrayType) ;
Label
  EndProc ;
Type
  ModeType       = 1 .. 3 ;            { 1:derivative sign ; 2:zero ; 3:inverse of derivative sign }
  ModeSetType    = Set Of ModeType ;
  TableEntryType = Record
                     Sign : SignType ;
                     Mode : ModeType
                   End ;
  TableType       = Array [EquationRange, ParameterRange] Of TableEntryType ;
  EndoInfoType    = Array [EquationRange] Of Record
                                             EndoNo : ParameterRangeW ;
                                             EndoAr : Array [ParameterRange] Of ParameterRange
                                           End ;
  ParEqnNoType    = Array [ParameterRange] Of EquationRangeW ;
   ExoSetType      = Set Of ParameterRange ;
Var
  Table         : TableType      ;
  Endo          : EndoInfoType   ;
  ParEqnNo      : ParEqnNoType   ;
  ExoSet        : ExoSetType     ;
  NoEqn         : EquationRangeW ;
  NoPar         : ParameterRange ;
  SignCh        : Char           ;
  PossibilityNo : Integer        ;
  Initialized ,
  InitOkey    ,
  Continue      : Boolean        ;

  Procedure TableEntry (Var Entry : TableEntryType ;
                            Sign  : SignType       ;
                            Mode  : ModeType       ) ;
  Begin
    Entry. Sign := Sign ;
    Entry. Mode := Mode
  End ;

  Procedure WritePossibility ;
  Var
    NoPar : ParameterRange ;
  Begin
    With CrModel. Model. Parameters Do
      For NoPar:=1 To ParNo Do
        If (Signs [NoPar] = _undef) Then
        Begin
          Write (TrimT (ParInfo [NoPar]. Name), ' = ') ;
          Case Table [ParEqnNo [NoPar], NoPar]. Sign Of
            neg_  : Write ('-') ;
            zero_ : Write ('0') ;
            pos_  : Write ('+')
          End ;
          Write ('   ')
        End ;

   WriteSc ('') ;

   PossibilityNo := PossibilityNo + 1
  End ;

  { *** finds next possible sign combinations using Table *** }
  Function NextPossibility (NoEqn : EquationRange) : Boolean ;
  Var
    NoPar     : ParameterRangeW ;
    i         : ParameterRange  ;
    ExitLoop,
    Found     : Boolean         ;

    { *** modes consistent if all modes are zero or if there are at least *** }
    { *** two parameters one with mode 1 and the other with mode 3        *** }
    Function ModesConsistent (NoEqn : EquationRange) : Boolean ;
    Var
      NoPar   : ParameterRange ;
      ModeSet : ModeSetType    ;
    Begin
      ModeSet := [] ;

      With CrModel, Model. Parameters Do
        For NoPar:=1 To ParNo Do
        If ParInEqn (Model, NoPar, NoEqn) Then
          ModeSet := ModeSet + [Table [NoEqn, NoPar]. Mode] ;

    ModesConsistent := ([1,3] * ModeSet = []) Or ([1,3] * ModeSet = [1,3])
  End ;

Begin
  ExitLoop := False ;

  With CrModel. ModelInfo, Endo [NoEqn] Do
    Repeat
      ExitLoop := True   ;
      Found    := False  ;
      NoPar    := EndoNo ;

      While (Not Found) And (NoPar > 0) Do
        If Table [NoEqn, EndoAr [NoPar]]. Mode < 3
          Then Found := True
          Else NoPar := NoPar - 1 ;

      If Found Then
      Begin
        Table [NoEqn, EndoAr [NoPar]]. Mode := Table [NoEqn, EndoAr [NoPar]]. Mode + 1 ;{ try next mode }
        Case Table [NoEqn, EndoAr [NoPar]]. Mode Of
          1 : Table [NoEqn, EndoAr [NoPar]]. Sign := IMatrixS [NoEqn, EndoAr [NoPar]] ;
          2 : Table [NoEqn, EndoAr [NoPar]]. Sign := zero_ ;
          3 : Table [NoEqn, EndoAr [NoPar]]. Sign := SignInverse (IMatrixS [NoEqn, EndoAr [NoPar]])
        End ;
        For i:=NoPar+1 To EndoNo Do                              { initialize following parameters }
          TableEntry (Table [NoEqn, EndoAr [i]], IMatrixS [NoEqn, EndoAr [i]], 1) ;
        If Not ModesConsistent (NoEqn) Then
          ExitLoop := False
      End
    Until ExitLoop ;

  NextPossibility := Found
End ;

{ *** initialize signs & modes of parameters in the equation *** }
Function InitializeEqn (NoEqn : EquationRange) : Boolean ;
Var
  NoPar   : ParameterRange ;
  ModeSet : ModeSetType    ;
  EqnOkey : Boolean        ;
Begin
  With CrModel, Model, ModelInfo, Equations. EqnInfo [NoEqn], Parameters, Endo [NoEqn] Do
  Begin
    ModeSet := [] ;

    For NoPar:=1 To ParNo Do
      If (ParInEqn (Model,NoPar,NoEqn)) And ((Signs [NoPar] <> _undef) Or (ParEqnNo[NoPar] < NoEqn)) Then
        ModeSet := ModeSet + [Table [NoEqn, NoPar]. Mode] ;

    EqnOkey := True ;

    Case EndoNo Of
      0  : If ([1,3] * ModeSet <> []) And ([1,3] * ModeSet <> [1,3]) Then      { so, equation cannot }
             EqnOkey := False ;                                               { be initialized      }
      1  : If ([1,3] * ModeSet = []) Then
             TableEntry (Table [NoEqn, EndoAr [1]], zero_, 2)
           Else
             If (3 IN ModeSet)
               Then TableEntry (Table [NoEqn, EndoAr [1]], IMatrixS [NoEqn, EndoAr [1]], 1)
               Else TableEntry (Table [NoEqn, EndoAr [1]], SignInverse (IMatrixS[NoEqn, EndoAr[1]]), 3) ;
     Else If (3 IN ModeSet) Then
             For NoPar:=1 To EndoNo Do
               TableEntry (Table [NoEqn, EndoAr [NoPar]], IMatrixS [NoEqn, EndoAr [NoPar]], 1)
           Else Begin
             For NoPar:=1 To EndoNo-1 Do
```

```
              TableEntry (Table [NoEqn, EndoAr [NoPar]], IMatrixS [NoEqn, EndoAr [NoPar]], 1) ;
             TableEntry (Table [NoEqn, EndoAr [EndoNo]], SignInverse (IMatrixS[NoEqn,EndoAr[EndoNo]]), 3)
           End
       End { Case }
     End ; { With }

     InitializeEqn := EqnOkey
    End ;

    Function InitialPossibility (NoEqn : EquationRange) : Boolean ;
     Var
      NoPar : ParameterRange ;
    Begin
      With CrModel, ModelInfo, Model. Parameters Do
        For NoPar:=1 To ParNo Do
          If (ParInEqn (Model, NoPar, NoEqn)) And (Signs [NoPar] = _undef) And (ParEqnNo[NoPar] < NoEqn) Then
          Begin
            Table [NoEqn, NoPar]. Sign := Table [ParEqnNo [NoPar], NoPar]. Sign ;
            If (Table [NoEqn, NoPar]. Sign = IMatrixS [NoEqn, NoPar]) Then
              Table [NoEqn, NoPar]. Mode := 1
            Else
              If Table [NoEqn, NoPar]. Sign = zero_
                Then Table [NoEqn, NoPar]. Mode := 2
                Else Table [NoEqn, NoPar]. Mode := 3
          End ;

      InitialPossibility := InitializeEqn (NoEqn)
    End ;

  Begin  { SignAnalysis }
    With CrModel, ModelInfo, Model, Equations, Parameters Do
    Begin
      ExoSet := [] ;

      For NoPar:=1 To ParNo Do
        If Signs [NoPar] <> _undef Then
          ExoSet := ExoSet + [NoPar] ;

      Writeln (LF, 'Possibilities',
               LF, '-------------' ) ;

      PossibilityNo := 0 ;

      If ExoSet = [1..ParNo] Then                      { if no unknowns then no possibility }
        Goto EndProc ;

      FillChar (ParEqnNo, ParNo, 0) ;

      For NoEqn:=1 To EqnNo Do
        With EqnInfo [NoEqn], Endo [NoEqn] Do
        Begin
          EndoNo := 0 ;

          For NoPar:=1 To ParNo Do
            If ParInEqn (Model, NoPar, NoEqn) Then
            Begin
              If Signs [NoPar] <> _undef Then         { assign signs and modes of known parameters into }
              Begin                                   { Table. These do not change during the analysis  }
                Table [NoEqn, NoPar]. Sign := SignType (Ord (Signs [NoPar])) ;
                If Ord (Signs [NoPar]) = Ord (IMatrixS [NoEqn, NoPar]) Then
                  Table [NoEqn, NoPar]. Mode := 1
                Else
                  If Signs [NoPar] = _zero
                    Then Table [NoEqn, NoPar]. Mode := 2
                    Else Table [NoEqn, NoPar]. Mode := 3
              End ;

              If Not (NoPar IN ExoSet) Then
              Begin
            EndoNo := EndoNo + 1 ;
            EndoAr [EndoNo] := NoPar ;
            ExoSet := ExoSet + [NoPar]
          End ;

          If ParEqnNo [NoPar] = 0 Then
            ParEqnNo [NoPar] := NoEqn              { # of the "first" equation the parameter appears }
        End { If & For }
    End ; { With & For }

    NoEqn := 1 ;
    Initialized := False ;
    Repeat
      InitOkey := InitializeEqn (NoEqn) ;
      If Endo [NoEqn]. EndoNo = 0 Then       { if # of endos in NoEqn > 0, then InitializeEqn returns True }
        If Not InitOkey
          Then Goto EndProc
          Else NoEqn := NoEqn + 1
      Else
        Initialized := True
    Until Initialized ;

    With Endo [NoEqn] Do
      Table [NoEqn, EndoAr [EndoNo]]. Mode := Table [NoEqn, EndoAr [EndoNo]]. Mode - 1 ;

    While (NoEqn > 0) Do
    Begin
      Continue := True ;
      If NextPossibility (NoEqn) Then
      Begin
        While (Continue) And (NoEqn < EqnNo) Do       { initialize following equations }
        Begin
          NoEqn := NoEqn + 1 ;
          If Not InitialPossibility (NoEqn) Then
          Begin
            NoEqn := NoEqn - 1 ;
            Continue := False
          End
        End ; { While }
        If Continue Then
        Begin
          WritePossibility ;
          NoEqn := EqnNo
        End
      End { If }
      Else
        NoEqn := NoEqn - 1
    End { While }
  End ; { With }

EndProc :

  Writeln ;
  If PossibilityNo = 0
    Then Write ('Inconsistent, no possibilities')
    Else Write ('Possibilities found : ', PossibilityNo)
End ;


{$I INCLUDE.PAS}


{ *** displays main menu and calls the option selected *** }
Procedure MainMenu ;           { no variables must be passed to this procedure }
Var
  Choice : Integer ;
Begin
  ClrScr ;
```

```
GotoXY (54, 3) ;
Write  ('Current model : ') ;
If CrModelGiven Then
  Write (CrModel. Model. Name) ;

GotoXY (1, 2) ;
Write ('Analysis of Algebraic Mathematical Models  ver.1.00', LF,
       'Main Menu', LF,
       '---------', LF,
       LF,
       '1.  New model (from user)', LF,
       '2.  New model (from library)', LF,
       '3.  Sign of Information matrix', LF,
       '4.  Sign analysis of total differentials', LF,
       '5.  Sign analysis of gains', LF,
       '6.  Parameter values', LF,
       '7.  Value analysis', LF,
       '8.  Gains', LF,
       '9.  Elasticities', LF,
       LF,
       '10. List library', LF,
       '11. Insert to library', LF,
       '12. Delete from library', LF,
       '13. Save current model', LF,
       LF,
       '14. Exit', LF,
       LF,
       'Choice : ') ;

ReadInputI (2, Choice) ;

If (Not CrModelGiven) And (Choice IN [3..9,13]) Then
  GiveError (8, 0) ;
If (Not IMatrixSGiven) And (Choice IN [4,5]) Then
  GiveError (9, 0) ;
If (Not ParValuesGiven) And (Choice IN [7..9]) Then
  GiveError (10, 0) ;
If Not (Choice IN [1..14]) Then
  GiveError (20, 0) ;

If (Choice <> 14) Then
Begin
  ClrScr ;
  Write ('Option ', Choice, ' - ') ;
  Case Choice Of
    1  : Write ('New model (from user)', LF,
                'A new model is given by the user and becomes the current model') ;
    2  : Write ('New model (from library)', LF,
                'A model of the library is loaded as the current model') ;
    3  : Write ('Sign of Information matrix', LF,
                'Signs of pars & par expressions given to reduce I matrix into a sign matrix') ;
    4  : Write ('Sign analysis of total differentials', LF,
                'Sign analysis of tot difs on current model, signs of ? pars are determined') ;
    5  : Write ('Sign analysis of gains', LF,
                'Sign analysis of gains on current model, gains of endos wrt a exo/cst determined') ;
    6  : Write ('Parameter values', LF,
                'Par values given by the user & equations solved, if desired, for endo vars') ;
    7  : Write ('Value analysis', LF,
                'Value analysis of tot difs on current model, magnitudes of ? pars are determined') ;
    8  : Write ('Gains', LF,
                'Display gains for a single exo var/constant or for all exo vars/constants') ;
    9  : Write ('Elasticities', LF,
                'Display elasticities for a single exo var/constant or for all exo vars/constants') ;
    10 : Write ('List library', LF,
                'Models in the library are listed') ;
    11 : Write ('Insert to library', LF,
                'Add a new model to the library') ;
    12 : Write ('Delete from library', LF,
                'Remove a model from the library') ;
```

```
      13 : Write ('Save current model', LF,
                  'Add the current model to the library')
    End ;

    GotoXY (54, 1) ;
    Write  ('Current model : ') ;
    If CrModelGiven Then
      Write (CrModel. Model. Name) ;

    Write  (LF, LF, '--------------------------------------------------------------------------------') ;
    GotoXY (1, 24) ;
    Write  ('--------------------------------------------------------------------------------') ;
    Window (1, 4, 80, 23) ;
    GotoXY (1, 1)
  End ;

  Case Choice Of
    1  : ReadCurrentModel ;
    2  : LoadCurrentModel ;
    3  : SignOfInformationMatrix ;
    4  : SignAnalysisOfTotalDifferentials ;
    5  : SignAnalysisOfGains ;
    6  : ParameterValues ;
    7  : ValueAnalysisOfTotalDifferentials ;
    8  : DisplayGains ;
    9  : Elasticities ;
    10 : ListLibrary ;
    11 : InsertToLibrary ;
    12 : DeleteFromLibrary ;
    13 : SaveCurrentModel ;
    14 : Begin
           Close (ModelLibFile) ;
           Halt
         End
  End { Case }
End ;


{ *** main program *** }
Begin
  Assign (ModelLibFile, ModelLibraryFileName) ;       { open model library file }

  {$I-}
  Reset  (ModelLibFile) ;
  {$I+}
  If (IOResult <> 0) Then                             { if file does not exist, create it }
    Rewrite (ModelLibFile) ;

  CrModelGiven := False ;                             { no currnt model }

  Inline ($89/$26/SPReg/         { MOV SPReg,SP }    { move (address of top of stack - 2) to SPReg      }
          $FF/$0E/SPReg/         { DEC SPReg    }    { this is the address that will contain the        }
          $FF/$0E/SPReg  ) ;     { DEC SPReg    }    { return address of main program when it calls     }
                                                     { the procedure MainMenu. MainMenu must not be     }
                                                     { called with a variable, otherwise SPReg points to }
                                                     { these variables instead of the return address    }

  HeapPtrSeg := Seg (HeapPtr^) ;              { top of heap: segment & offset. The program allocates   }
  HeapPtrOfs := Ofs (HeapPtr^) ;              { pointers on heap at top of heap & at (top of heap + 32K) }

  TextColor (LightGray) ;

  Repeat

    MainMenu

  Until False
End.
```

```
{ *** sign analysis of total differentials *** }
Procedure SignAnalysisOfTotalDifferentials ;
Var
  Signs  : SignArrayType  ;
  NoPar  : ParameterRange ;
  SignCh : Char           ;
Begin
  With CrModel. Model. Parameters Do
    For NoPar:=1 To ParNo Do
    Begin
      Write (ParInfo [NoPar]. Name, ' (- 0 + ?) : ') ;
      ReadChar (SignCh, ['-','0','+','?']) ;
      Case SignCh Of
        '-' : Signs [NoPar] := _neg   ;
        '0' : Signs [NoPar] := _zero  ;
        '+' : Signs [NoPar] := _pos   ;
        '?' : Signs [NoPar] := _undef
      End
    End ;

  SignAnalysis (Signs) ;

  Message (3)
End ;

{ *** sign analysis of gains. similar to sign analysis of total  *** }
{ *** differentials. sign of given exo/const becomes +. signs of *** }
{ *** other exos/consts initialized. signs of endos found        *** }
Procedure SignAnalysisOfGains ;
Var
  Signs   : SignArrayType  ;
  ParName : ParameterName  ;
  Index ,
  NoPar   : ParameterRange ;
  SignCh  : Char           ;
Begin
  With CrModel, Model, Parameters Do
  Begin
    Write ('Name of exo var/constant : ') ;
    ReadInputS (ParameterNameLen, ParName) ;

    If Not ParameterExists (Parameters, ParName, Index) Then
      GiveError (24, 0) ;

    If ParInfo [Index]. Type_ = Endo Then
      GiveError (25, 0) ;

    Writeln ;

    For NoPar:=1 To ParNo Do
      If ParInfo [NoPar]. Type_ = Endo Then
        Signs [NoPar] := _undef
      Else
        If ParInfo [NoPar]. Name = ParName Then
          Signs [NoPar] := _pos
        Else Begin
          Write (ParInfo [NoPar]. Name, ' (- 0 +) : ') ;
          ReadChar (SignCh, ['-','0','+']) ;
          Case SignCh Of
            '-' : Signs [NoPar] := _neg  ;
            '0' : Signs [NoPar] := _zero ;
            '+' : Signs [NoPar] := _pos
          End
        End ;

    SignAnalysis (Signs)
  End ; { With }
  Message (10)
End ;

{ *** input a system state *** }
Procedure ParameterValues ;
Var
  EqnVector      : EquationVectorType ;
  EqnMatrix      : EquationMatrixType ;
  NoEqn          : EquationRange      ;
  NoPar          : ParameterRange     ;
  Answer         : Char               ;

  { *** solves the nonlinear equations by Newton-Raphson method *** }
  Procedure Newton_Raphson ;
  Const
    Tolerance    = 1E-6 ;
    MaxIteration = 10   ;                { 10 iterations is enough for convergence }
  Var
    EqnVector      : EquationVectorType ;
    EqnMatrix      : EquationMatrixType ;
    NoEqn          : EquationRange      ;
    NoPar          : ParameterRange     ;
    OldVector ,
    NewVector ,
    RHSVector      : EquationVectorType  ;
    IterationNo    : Byte                ;
    Value1 ,
    Value2         : Real                ;
  Begin
    IterationNo := 0 ;

    With CrModel, ModelInfo, Model, Equations, Parameters Do
      Repeat
        For NoEqn:=1 To EqnNo Do
          EqnVector [NoEqn] := EvaluateExpression (EqnInfo [NoEqn]. Expression) ;

        For NoEqn:=1 To EqnNo Do
          For NoPar:=1 To EndoParNo Do
            EqnMatrix [NoEqn,NoPar] := EvaluateExpression (EqnInfo[NoEqn]. IMatrixRow [EndoParAr[NoPar]]) ;

        For NoPar:=1 To EndoParNo Do
          OldVector [NoPar] := Value [EndoParAr [NoPar]] ;

        MatrixVectorMult (EqnNo, EqnMatrix, OldVector, RHSVector) ;

        For NoEqn:=1 To EqnNo Do
          RHSVector [NoEqn] := RHSVector [NoEqn] - EqnVector [NoEqn] ;

        GaussElimination (EqnMatrix, RHSVector, EqnNo, NewVector) ;

        For NoPar:=1 To EndoParNo Do
          Value [EndoParAr [NoPar]] := NewVector [NoPar] ;

        Value1 := 0 ;
        Value2 := 0 ;

        For NoPar:=1 To EndoParNo Do
          Value1 := Value1 + Sqr (NewVector [NoPar] - OldVector [NoPar]) ;

        For NoEqn:=1 To EqnNo Do
          Value2 := Value2 + Sqr (EvaluateExpression (EqnInfo [NoEqn]. Expression)) ;

        IterationNo := IterationNo + 1
      Until ((Sqrt (Value1) < Tolerance) And (Sqrt (Value2) < Tolerance)) Or (IterationNo > MaxIteration) ;

    If IterationNo > MaxIteration Then
      GiveError (11, 0)
  End ;
```

```
Begin
  ParValuesGiven := False ;

  With CrModel, ModelInfo, Model, Equations, Parameters Do
  Begin
    For NoPar:=1 To ParNo Do
    Begin
      Write (ParInfo [NoPar]. Name, ' = ') ;
      ReadInputR (10, Value [NoPar])
    End ;

    Write (LF, 'Will the equations be solved for endogenous variables ? (Y/N) ') ;
    ReadChar (Answer, ['Y','N']) ;

    Writeln ;

    If Answer = 'Y' Then
    Begin
      Newton_Raphson ;
      For NoPar:=1 To EndoParNo Do
      Begin
        Write   (ParInfo [EndoParAr [NoPar]]. Name, ' = ', Value [EndoParAr [NoPar]] :7:2) ;
        WriteSc ('')
      End
    End ;

    For NoEqn:=1 To EqnNo Do
      For NoPar:=1 To EndoParNo Do
        EqnMatrix [NoEqn, NoPar] := EvaluateExpression (EqnInfo [NoEqn]. IMatrixRow [EndoParAr [NoPar]]) ;

    For NoPar:=1 To ExoParNo Do        { calculate gains }
    Begin
      For NoEqn:=1 To EqnNo Do
      Begin
        EqnVector [NoEqn] := EvaluateExpression (EqnInfo [NoEqn]. IMatrixRow [ExoParAr [NoPar]]) ;
        EqnVector [NoEqn] := - EqnVector [NoEqn]
      End ;
      GaussElimination (EqnMatrix, EqnVector, EqnNo, ExoEndoMatrix [NoPar])
    End
  End ; { With }

  Message (2) ;

  ParValuesGiven := True
End ;

{ *** value analysis on the current model *** }
Procedure ValueAnalysisOfTotalDifferentials ;
Type
  MagnitudeArrayType = Array [ParameterRange] Of Real ;
Var
  Signs        : SignArrayType      ;
  Magnitudes   : MagnitudeArrayType ;
  EqnVector ,
  ResultVector : EquationVectorType ;
  EqnMatrix    : EquationMatrixType ;
  Gain         : Real               ;
  NoEqn        : EquationRange      ;
  UndefNo ,
  NoPar ,
  i            : ParameterRange     ;
  SignCh       : Char               ;
Begin
  With CrModel, ModelInfo, Model, Equations, Parameters Do
  Begin
    UndefNo := 0 ;

    Writeln ('              Sign (- 0 + ?)  Magnitude') ;
    Writeln ('              --------------  ---------') ;
```

```
For NoPar:=1 To ParNo Do
Begin
  Write (ParInfo [NoPar]. Name, ' :       ') ;
  ReadChar (SignCh, ['-','0','+','?']) ;
  Case SignCh Of
    '-' : Signs [NoPar] := _neg ;
    '0' : Begin
            Signs [NoPar] := _zero ;
            Magnitudes [NoPar] := 0
          End ;
    '+' : Signs [NoPar] := _pos ;
    '?' : Begin
            Signs [NoPar] := _undef ;
            UndefNo := UndefNo + 1
          End
  End ;
  If Signs [NoPar] IN [_neg, _pos] Then
  Begin
    GotoXY (33, WhereY-1) ;
    ReadInputR (5, Magnitudes [NoPar]) ;
  End
End ;

If UndefNo <> EqnNo Then
  GiveError (26, 0) ;

For NoEqn:=1 To EqnNo Do          { solve the equations }
Begin
  EqnVector [NoEqn] := 0 ;
  UndefNo := 1 ;

  For NoPar:=1 To ParNo Do
  Begin
    If ParInfo [NoPar]. Type  = Endo Then
      If NoPar = EndoParAr [NoEqn]
        Then Gain := 1
        Else Gain := 0
    Else Begin
      i := 1 ;
      While ExoParAr [i] <> NoPar Do
        i := i + 1 ;
      Gain := - ExoEndoMatrix [i, NoEqn]
    End ;

    If Signs [NoPar] = _undef Then
    Begin
      EqnMatrix [NoEqn, UndefNo] := Gain ;
      UndefNo := UndefNo + 1
    End
    Else Begin
      If Signs [NoPar] = _pos Then
        Gain := - Gain ;
      EqnVector [NoEqn] := EqnVector [NoEqn] + Gain * Magnitudes [NoPar]
    End
  End { For NoPar ... Do Begin }
End ; { For NoEqn ... Do Begin }

GaussElimination (EqnMatrix, EqnVector, EqnNo, ResultVector) ;

Write (LF,
       'Results', LF,
       '-------', LF ) ;

i := 1 ;
For NoPar:=1 To ParNo Do
  If Signs [NoPar] = _undef Then
  Begin
    Writeln (ParInfo [NoPar]. Name, ' = ', ResultVector [i] :6:2) ;
```

```
        i := i + 1
      End
    End ; { With }

    Message (11)
  End ;

  { *** output gains for the given system state *** }
  Procedure DisplayGains ;
  Var
    ParName  : ParameterName  ;
    Index ,
    NoPar ,
    i        : ParameterRange ;
    AllGains : Boolean        ;
  Begin
    With CrModel, ModelInfo, Model, Parameters Do
    Begin
      Write ('Name of exo var/constant : ') ;
      ReadInputS (ParameterNameLen, ParName) ;

      AllGains := (TrimT (ParName) = '') ;           { if no name is given then output all gains }

      If (Not AllGains) Then
        If Not ParameterExists (Parameters, ParName, Index) Then
          GiveError (24, 0)
        Else
          If ParInfo [Index]. Type_ = Endo Then
            GiveError (25, 0) ;

      Writeln ;

      For NoPar:=1 To ExoParNo Do
        If AllGains Or (ExoParAr [NoPar] = Index) Then
        Begin
          Writeln ('Gains for ', ParInfo [ExoParAr [NoPar]]. Name) ;
          WriteSc (Spc ('-', Length (TrimT (ParInfo [ExoParAr [NoPar]]. Name)) + 10)) ;
          For i:=1 To EndoParNo Do
            Writeln (ParInfo [EndoParAr [i]]. Name, ' : ', ExoEndoMatrix [NoPar, i] :6:2)
        End
    End ; { With }

    Message (12)
  End ;

  { *** output elasticities for the given system state *** }
  Procedure Elasticities ;
  Var
    ParName         : ParameterName  ;
    Index ,
    NoPar ,
    i               : ParameterRange ;
    AllElasticities : Boolean        ;
  Begin
    With CrModel, ModelInfo, Model, Parameters Do
    Begin
      Write ('Name of exo var/constant : ') ;
      ReadInputS (ParameterNameLen, ParName) ;

      AllElasticities := (TrimT (ParName) = '') ;      { if no name is given then output all elasticities }

      If (Not AllElasticities) Then
        If Not ParameterExists (Parameters, ParName, Index) Then
          GiveError (24, 0)
        Else
          If ParInfo [Index]. Type_ = Endo Then
            GiveError (25, 0) ;

      Writeln ;
    For NoPar:=1 To ExoParNo Do
      If AllElasticities Or (ExoParAr [NoPar] = Index) Then
      Begin
        Writeln ('Elasticities for ', ParInfo [ExoParAr [NoPar]]. Name) ;
        WriteSc (Spc ('-', Length (TrimT (ParInfo [ExoParAr [NoPar]]. Name)) + 17)) ;
        For i:=1 To EndoParNo Do
          Writeln (ParInfo [EndoParAr [i]]. Name, ' : ',
                   ExoEndoMatrix [NoPar, i] * Value [ExoParAr [NoPar]] / Value [EndoParAr [i]] :6:2)
      End
  End ; { With }

  Message (13)
End ;

{ *** simplify the expression. For example, if there is a + operation *** }
{ *** where both operands are numbers, then add those numbers and     *** }
{ *** insert the result as a single entry instead of three entries    *** }
Procedure SimplifyExpression (Var Expression : PostfixType) ;
Const
  MaxStackItem = 50 ;
Type
  StackItemRange  = 1 .. MaxStackItem ;
  StackItemRangeW = 0 .. MaxStackItem ;
  StackType       = Record
                      Top  : StackItemRangeW ;
                      Item : Array [StackItemRange] Of PostfixType
                    End ;
Var
  Stack           : StackType          ;
  NoEntry         : PostfixEntryRange  ;
  Item1, Item2,
  ItemA, ItemB,
  ItemC, ItemD    : PostfixType        ;
  Error           : Boolean            ;

  Procedure Pop (Var Stack   : StackType   ;
                 Var TopItem : PostfixType ) ;
  Begin
    With Stack Do
    Begin
      TopItem := Item [Top] ;
      Top := Top - 1
    End
  End ;

  Procedure Push (Var Stack   : StackType   ;
                      TopItem : PostfixType ) ;
  Begin
    With Stack Do
    Begin
      Top := Top + 1 ;
      Item [Top] := TopItem
    End
  End ;

  Function Result (Operand1 ,
                   Operand2  : Real         ;
                   Operator  : OperatorType ) : Real ;
  Begin
    If ((Operator = Div_  ) And (Operand2  = 0)) Or
       ((Operator = Ln_   ) And (Operand2 <= 0)) Or
       ((Operator = Log_  ) And (Operand2 <= 0)) Or
       ((Operator = Sqrt_ ) And (Operand2 <  0)) Or
       ((Operator = Power_) And
        ((Operand1 <= 0) And
        ((Operand1 >= 0) Or (Operand2 <> Int (Operand2))) And ((Operand1 <> 0) Or (Operand2 <= 0))))
      Then Error := True
      Else Result := OpResult (Operand1, Operand2, Operator)
```

```
 End ;

 Procedure NegativeOfItem (Var Item1 : PostfixType ;
                               Item2 : PostfixType  ) ;
 Begin
  Item1 := Item2 ;

  With Item1 Do
  Begin
   EntryNo := EntryNo + 1 ;
   Entry [EntryNo]. Type_    := Operator_ ;
   Entry [EntryNo]. Operator := UMinus_
  End
 End ;

 Function CondFalse (Condition : Boolean     ;
                     Item      : PostfixType  ) : Boolean ;
 Begin
  Condition := Condition And (Not Error) ;

  If Condition Then
   Push (Stack, Item) ;

  CondFalse := Not Condition
 End ;

 Procedure MergeItems (Item1, Item2 : PostfixType  ;
                       Operator     : OperatorType  ) ;
 Var
  i : PostfixEntryRange ;
 Begin
  For i:=1 To Item2. EntryNo Do
   Item1. Entry [Item1. EntryNo + i] := Item2. Entry [i] ;
  Item1. EntryNo := Item1. EntryNo + Item2. EntryNo + 1 ;
  Item1. Entry [Item1. EntryNo]. Type_    := Operator_ ;
  Item1. Entry [Item1. EntryNo]. Operator := Operator  ;

  Push (Stack, Item1)
 End ;

Begin  { SimplifyExpression }
 With Stack, Expression Do
 Begin
  Top := 0 ;

  For NoEntry:=1 To EntryNo Do
   If Entry [NoEntry]. Type_ <> Operator_ Then
   Begin
    ItemA. EntryNo := 1 ;
    ItemA. Entry [1] := Entry [NoEntry] ;
    Push (Stack, ItemA)
   End
   Else Begin
    Pop (Stack, Item2) ;
    If BinaryOperator (Entry [NoEntry]. Operator) Then
     Pop (Stack, Item1) ;

    Error := False ;

    Case Entry [NoEntry]. Operator Of
     Plus_   : Begin
                NumberEntry (ItemA, Item1. Entry [1]. Number + Item2. Entry [1]. Number) ;
                If CondFalse ((Item1. EntryNo = 1) And (Item1. Entry [1]. Type_ = Number_) And
                              (Item2.EntryNo = 1) And (Item2.Entry[1]. Type_ = Number_), ItemA) Then
                 If CondFalse ((Item1. EntryNo = 1) And (Item1. Entry [1]. Type_ = Number_) And
                                (Item1. Entry [1]. Number = 0), Item2) Then
                  If CondFalse ((Item2. EntryNo = 1) And (Item2. Entry [1]. Type_ = Number_) And
                                 (Item2. Entry [1]. Number = 0), Item1) Then
                   MergeItems (Item1, Item2, Plus_)
                End ;
     BMinus_ : Begin
                  NumberEntry (ItemA, Item1. Entry [1]. Number - Item2. Entry [1]. Number) ;
                  NegativeOfItem (ItemB, Item2) ;
                  If CondFalse ((Item1. EntryNo = 1) And (Item1. Entry [1]. Type_ = Number_) And
                                (Item2.EntryNo = 1) And (Item2. Entry[1].Type_ = Number_) , ItemA) Then
                    If CondFalse ((Item1. EntryNo = 1) And (Item1. Entry [1]. Type_ = Number_) And
                                  (Item1. Entry [1]. Number = 0), ItemB) Then
                      If CondFalse ((Item2. EntryNo = 1) And (Item2. Entry [1]. Type_ = Number_) And
                                    (Item2. Entry [1]. Number = 0), Item1) Then
                        MergeItems (Item1, Item2, BMinus_)
                End ;
     UMinus_, Sin_, Cos_, ArcTan_, Exp_, Ln_, Log_, Sqr_, Sqrt_ :
                Begin
                  NumberEntry (ItemA, Result (0, Item2. Entry [1]. Number, Entry [NoEntry]. Operator)) ;
                  Item1. EntryNo := 0 ;
                  If CondFalse ((Item2. EntryNo = 1) And (Item2.Entry[1].Type_ = Number_) , ItemA) Then
                    MergeItems (Item2, Item1, Entry [NoEntry]. Operator)
                End ;
     Mult_   : Begin
                  NumberEntry (ItemA, Item1. Entry [1]. Number * Item2. Entry [1]. Number) ;
                  NumberEntry (ItemB, 0) ;
                  NegativeOfItem (ItemC, Item2) ;
                  NegativeOfItem (ItemD, Item1) ;
                  If CondFalse ((Item1. EntryNo = 1) And (Item1. Entry [1]. Type_ = Number_) And
                                (Item2. EntryNo = 1) And (Item2.Entry[1].Type_ = Number_) , ItemA) Then
                    If CondFalse (((Item1. EntryNo = 1) And (Item1. Entry [1]. Type_ = Number_) And
                                   (Item1. Entry [1]. Number = 0)) Or
                                  ((Item2. EntryNo = 1) And (Item2. Entry [1]. Type_ = Number_) And
                                   (Item2. Entry [1]. Number = 0)), ItemB) Then
                      If CondFalse ((Item1. EntryNo = 1) And (Item1. Entry [1]. Type_ = Number_) And
                                    (Item1. Entry [1]. Number = 1), Item2) Then
                        If CondFalse ((Item2. EntryNo = 1) And (Item2. Entry [1]. Type_ = Number_) And
                                      (Item2. Entry [1]. Number = 1), Item1) Then
                          If CondFalse ((Item1. EntryNo = 1) And (Item1. Entry[1]. Type_ = Number_) And
                                        (Item1. Entry [1]. Number = -1), ItemC) Then
                            If CondFalse ((Item2. EntryNo = 1) And (Item2.Entry[1].Type_ = Number_) And
                                          (Item2. Entry [1]. Number = -1), ItemD) Then
                              MergeItems (Item1, Item2, Mult_)
                End ;
     Div_    : Begin
                  NumberEntry (ItemA, Result (Item1. Entry [1]. Number, Item2.Entry[1].Number, Div_)) ;
                  NumberEntry (ItemB, 0) ;
                  NegativeOfItem (ItemC, Item1) ;
                  If CondFalse ((Item1. EntryNo = 1) And (Item1. Entry [1]. Type_ = Number_) And
                                (Item2. EntryNo = 1) And (Item2.Entry[1].Type_ = Number_) , ItemA) Then
                    If CondFalse ((Item1. EntryNo = 1) And (Item1. Entry [1]. Type_ = Number_) And
                                  (Item1. Entry [1]. Number = 0), ItemB) Then
                      If CondFalse ((Item2. EntryNo = 1) And (Item2. Entry [1]. Type_ = Number_) And
                                    (Item2. Entry [1]. Number = 1), Item1) Then
                        If CondFalse ((Item2. EntryNo = 1) And (Item2. Entry [1]. Type_ = Number_) And
                                      (Item2. Entry [1]. Number = -1), ItemC) Then
                          MergeItems (Item1, Item2, Div_)
                End ;
     Power_  : Begin
                  NumberEntry (ItemA, 1) ;
                  NumberEntry (ItemB, 0) ;
                  NumberEntry (ItemC, Result (Item1. Entry[1].Number, Item2.Entry[1].Number, Power_)) ;
                  If CondFalse (((Item1. EntryNo = 1) And (Item1. Entry [1]. Type_ = Number_) And
                                 (Item1. Entry [1]. Number = 1)) Or
                                ((Item2. EntryNo = 1) And (Item2. Entry [1]. Type_ = Number_) And
                                 (Item2. Entry [1]. Number = 0)), ItemA) Then
                    If CondFalse ((Item1. EntryNo = 1) And (Item1. Entry [1]. Type_ = Number_) And
                                  (Item1. Entry [1]. Number = 0), ItemB) Then
                      If CondFalse ((Item2. EntryNo = 1) And (Item2. Entry [1]. Type_ = Number_) And
                                    (Item2. Entry [1]. Number = 1), Item1) Then
                        If CondFalse ((Item1. EntryNo = 1) And (Item1. Entry [1]. Type_ = Number_) And
                                      (Item2.EntryNo=1) And (Item2.Entry[1].Type_=Number_), ItemC) Then
                          MergeItems (Item1, Item2, Power_)
                  End ;
        End { Case }
      End ; { Else Begin & For }

    Expression := Item [1]

  End { With }
End ;

{ *** finds derivative of the equation wrt all parameters *** }
Procedure DerivativeOfEquation (Var Model : ModelType     ;
                                    NoEqn : EquationRange  ) ;

Const
  MaxDerivExpr = 50 ;
Type
  DerivExprRange = 1 .. MaxDerivExpr ;
  DerivExprType  = Record
                     Expr  : PostfixType ;
                     Deriv : Boolean
                   End ;
  DerivativeType = Record
                     ExprNo : DerivExprRange ;
                     ExprAr : Array [DerivExprRange] Of DerivExprType
                   End ;
Var
  Derivative : DerivativeType     ;
  NoExpr     : DerivExprRange     ;
  NoEntry    : PostfixEntryRangeW ;
  NoPar      : ParameterRange     ;
  Counter    : Integer            ;
  Found      : Boolean            ;

  Procedure ShiftExpressions (Var Derivative : DerivativeType ;
                                  Index      : DerivExprRange ;
                                  ShiftNo    : Byte            ) ;
  Var
    i : DerivExprRange ;
  Begin
    With Derivative Do
    Begin
      ExprNo := ExprNo + ShiftNo ;

      For i:=ExprNo DownTo Index+ShiftNo Do
        ExprAr [i] := ExprAr [i-ShiftNo]
    End
  End ;

  Procedure ShiftEntries (Var EntryAr : PostfixType ;
                              ShiftNo : Byte         ) ;
  Var
    i : PostfixEntryRange ;
  Begin
    With EntryAr Do
    Begin
      EntryNo := EntryNo - ShiftNo - 1 ;

      For i:=1 To EntryNo Do
        Entry [i] := Entry [i+ShiftNo]
    End
  End ;

  Procedure OperatorItem (Var Expression : DerivExprType ;
                              Ope        : OperatorType   ) ;
  Begin
    OperatorEntry (Expression. Expr, Ope) ;
    Expression. Deriv := False
  End ;
```

```
   Procedure NumberItem (Var Expression : DerivExprType ;
                             Number     : Real          ) ;
   Begin
     NumberEntry (Expression. Expr, Number) ;
     Expression. Deriv := False
   End ;

 Begin  { DerivativeOfEquation }
   With Derivative Do
   Begin
     ExprNo := 1 ;
     ExprAr [1]. Expr  := Model. Equations. EqnInfo [NoEqn]. Expression ;
     ExprAr [1]. Deriv := True ;

     Repeat
       Found  := False ;
       NoExpr := 1 ;

       While (NoExpr <= ExprNo) And (Not Found) Do
         If (ExprAr [NoExpr]. Deriv) And (ExprAr [NoExpr]. Expr. EntryNo > 1)
           Then Found  := True
           Else NoExpr := NoExpr + 1 ;

       If Found Then
       Begin
         With ExprAr [NoExpr]. Expr Do
         Begin
           NoEntry := EntryNo - 1 ;
           Counter := 1 ;
           While (Counter > 0) Do                        { find operator with lowest precedence }
           Begin
             If Entry [NoEntry]. Type_ <> Operator_ Then
               Counter := Counter - 1
             Else
               If BinaryOperator (Entry [NoEntry]. Operator) Then
                 Counter := Counter + 1 ;
             NoEntry := NoEntry - 1
           End
         End ; { With }

         Case ExprAr [NoExpr]. Expr. Entry [ExprAr [NoExpr]. Expr. EntryNo]. Operator Of
           Plus_ ,
           BMinus_  : Begin
                        ShiftExpressions (Derivative, NoExpr, 2) ;
                        ExprAr [NoExpr+1] := ExprAr [NoExpr] ;
                        ExprAr [NoExpr]. Expr. EntryNo := NoEntry ;
                        ShiftEntries (ExprAr [NoExpr+1]. Expr, NoEntry) ;
                        OperatorItem (ExprAr [NoExpr+2],
                                      ExprAr[NoExpr+2].Expr. Entry[ExprAr[NoExpr+2].Expr.EntryNo]. Operator)
                      End ;
           UMinus_  : Begin
                        ShiftExpressions (Derivative, NoExpr, 1) ;
                        ExprAr [NoExpr]. Expr. EntryNo := ExprAr [NoExpr]. Expr. EntryNo - 1 ;
                        OperatorItem (ExprAr [NoExpr+1], UMinus_)
                      End ;
           Mult_    : Begin
                        ShiftExpressions (Derivative, NoExpr, 6) ;
                        ExprAr [NoExpr+1] := ExprAr [NoExpr] ;
                        ExprAr [NoExpr]. Expr. EntryNo := NoEntry ;
                        ShiftEntries (ExprAr [NoExpr+1]. Expr, NoEntry) ;
                        ExprAr [NoExpr+3] := ExprAr [NoExpr] ;
                        ExprAr [NoExpr+3]. Deriv := False ;
                        ExprAr [NoExpr+4] := ExprAr [NoExpr+1] ;
                        ExprAr [NoExpr+1]. Deriv := False ;
                        OperatorItem (ExprAr [NoExpr+2], Mult_) ;
                        OperatorItem (ExprAr [NoExpr+5], Mult_) ;
                        OperatorItem (ExprAr [NoExpr+6], Plus_)
                      End ;
           Div_     : Begin
```

```
                ShiftExpressions (Derivative, NoExpr, 9) ;
                ExprAr [NoExpr+1] := ExprAr [NoExpr] ;
                ExprAr [NoExpr]. Expr. EntryNo := NoEntry ;
                ShiftEntries (ExprAr [NoExpr+1]. Expr, NoEntry) ;
                ExprAr [NoExpr+3] := ExprAr [NoExpr] ;
                ExprAr [NoExpr+3]. Deriv := False ;
                ExprAr [NoExpr+4] := ExprAr [NoExpr+1] ;
                ExprAr [NoExpr+1]. Deriv := False ;
                ExprAr [NoExpr+7] := ExprAr [NoExpr+1] ;
                OperatorItem (ExprAr [NoExpr+2], Mult_  ) ;
                OperatorItem (ExprAr [NoExpr+5], Mult_  ) ;
                OperatorItem (ExprAr [NoExpr+6], BMinus_) ;
                OperatorItem (ExprAr [NoExpr+8], Sqr_   ) ;
                OperatorItem (ExprAr [NoExpr+9], Div_   )
              End ;
Power_   : Begin
                ShiftExpressions (Derivative, NoExpr, 13) ;
                ExprAr [NoExpr+1] := ExprAr [NoExpr] ;
                ExprAr [NoExpr]. Expr. EntryNo := NoEntry ;
                ShiftEntries (ExprAr [NoExpr+1]. Expr, NoEntry) ;
                ExprAr [NoExpr+8] := ExprAr [NoExpr] ;
                ExprAr [NoExpr+3] := ExprAr [NoExpr+1] ;
                ExprAr [NoExpr  ]. Deriv := False ;
                ExprAr [NoExpr+1]. Deriv := False ;
                ExprAr [NoExpr+4] := ExprAr [NoExpr] ;
                ExprAr [NoExpr+9] := ExprAr [NoExpr] ;
                ExprAr [NoExpr+7] := ExprAr [NoExpr+1] ;
                OperatorItem (ExprAr [NoExpr+ 2], Power_) ;
                OperatorItem (ExprAr [NoExpr+ 5], Ln_   ) ;
                OperatorItem (ExprAr [NoExpr+ 6], Mult_ ) ;
                OperatorItem (ExprAr [NoExpr+10], Div_  ) ;
                OperatorItem (ExprAr [NoExpr+11], Mult_ ) ;
                OperatorItem (ExprAr [NoExpr+12], Plus_ ) ;
                OperatorItem (ExprAr [NoExpr+13], Mult_ )
              End ;
Sin_     : Begin
               ShiftExpressions (Derivative, NoExpr, 3) ;
               ExprAr [NoExpr]. Expr. EntryNo := ExprAr [NoExpr]. Expr. EntryNo - 1 ;
               ExprAr [NoExpr+2] := ExprAr [NoExpr] ;
               ExprAr [NoExpr]. Deriv := False ;
               OperatorItem (ExprAr [NoExpr+1], Cos_ ) ;
               OperatorItem (ExprAr [NoExpr+3], Mult_)
             End ;
Cos_     : Begin
               ShiftExpressions (Derivative, NoExpr, 4) ;
               ExprAr [NoExpr]. Expr. EntryNo := ExprAr [NoExpr]. Expr. EntryNo - 1 ;
               ExprAr [NoExpr+2] := ExprAr [NoExpr] ;
               ExprAr [NoExpr]. Deriv := False ;
               OperatorItem (ExprAr [NoExpr+1], Sin_   ) ;
               OperatorItem (ExprAr [NoExpr+3], Mult_  ) ;
               OperatorItem (ExprAr [NoExpr+4], UMinus_)
             End ;
ArcTan_  : Begin
               ShiftExpressions (Derivative, NoExpr, 5) ;
               ExprAr [NoExpr]. Expr. EntryNo := ExprAr [NoExpr]. Expr. EntryNo - 1 ;
               ExprAr [NoExpr+2] := ExprAr [NoExpr] ;
               ExprAr [NoExpr+2]. Deriv := False ;
               NumberItem   (ExprAr [NoExpr+1], 1    ) ;
               OperatorItem (ExprAr [NoExpr+3], Sqr_ ) ;
               OperatorItem (ExprAr [NoExpr+4], Plus_) ;
               OperatorItem (ExprAr [NoExpr+5], Div_ )
             End ;
Exp_     : Begin
               ShiftExpressions (Derivative, NoExpr, 3) ;
               ExprAr [NoExpr]. Expr. EntryNo := ExprAr [NoExpr]. Expr. EntryNo - 1 ;
               ExprAr [NoExpr+2] := ExprAr [NoExpr] ;
               ExprAr [NoExpr]. Deriv := False ;
               OperatorItem (ExprAr [NoExpr+1], Exp_ ) ;
               OperatorItem (ExprAr [NoExpr+3], Mult_)
```

```
                    End ;
           Ln_     : Begin
                       ShiftExpressions (Derivative, NoExpr, 2) ;
                       ExprAr [NoExpr]. Expr. EntryNo := ExprAr [NoExpr]. Expr. EntryNo - 1 ;
                       ExprAr [NoExpr+1] := ExprAr [NoExpr] ;
                       ExprAr [NoExpr+1]. Deriv := False ;
                       OperatorItem (ExprAr [NoExpr+2], Div_)
                     End ;
           Log_    : Begin
                       ShiftExpressions (Derivative, NoExpr, 4) ;
                       ExprAr [NoExpr]. Expr. EntryNo := ExprAr [NoExpr]. Expr. EntryNo - 1 ;
                       ExprAr [NoExpr+1] := ExprAr [NoExpr] ;
                       ExprAr [NoExpr+1]. Deriv := False ;
                       OperatorItem (ExprAr [NoExpr+2], Div_  ) ;
                       NumberItem   (ExprAr [NoExpr+3], LogOfe) ;
                       OperatorItem (ExprAr [NoExpr+4], Mult_ )
                     End ;
           Sqr_    : Begin
                       ShiftExpressions (Derivative, NoExpr, 4) ;
                       ExprAr [NoExpr]. Expr. EntryNo := ExprAr [NoExpr]. Expr. EntryNo - 1 ;
                       ExprAr [NoExpr+3] := ExprAr [NoExpr] ;
                       ExprAr [NoExpr]. Deriv := False ;
                       NumberItem   (ExprAr [NoExpr+1], 2    ) ;
                       OperatorItem (ExprAr [NoExpr+2], Mult_) ;
                       OperatorItem (ExprAr [NoExpr+4], Mult_)
                     End ;
           Sqrt_   : Begin
                       ShiftExpressions (Derivative, NoExpr, 5) ;
                       ExprAr [NoExpr]. Expr. EntryNo := ExprAr [NoExpr]. Expr. EntryNo - 1 ;
                       ExprAr [NoExpr+2] := ExprAr [NoExpr] ;
                       ExprAr [NoExpr+2]. Deriv := False ;
                       NumberItem   (ExprAr [NoExpr+1], 2    ) ;
                       OperatorItem (ExprAr [NoExpr+3], Sqrt_) ;
                       OperatorItem (ExprAr [NoExpr+4], Mult_) ;
                       OperatorItem (ExprAr [NoExpr+5], Div_ )
                     End
         End { Case }
       End { If Found Then Begin }
     Until Not Found ;

     With Model, Equations, EqnInfo [NoEqn], Parameters Do
       For NoPar:=1 To ParNo Do
         If ParInEqn (Model, NoPar, NoEqn) Then
           With IMatrixRow [NoPar] Do
           Begin
             EntryNo := 0 ;

             For NoExpr:=1 To ExprNo Do
               If ExprAr [NoExpr]. Deriv Then
               Begin
                 EntryNo := EntryNo + 1 ;
                 Entry [EntryNo]. Type_ := Number_ ;
                 If ExprAr [NoExpr]. Expr. Entry [1]. Index = NoPar
                   Then Entry [EntryNo]. Number := 1
                   Else Entry [EntryNo]. Number := 0
               End
               Else Begin
                 For NoEntry:=1 To ExprAr [NoExpr]. Expr. EntryNo Do
                   Entry [EntryNo + NoEntry] := ExprAr [NoExpr]. Expr. Entry [NoEntry] ;
                 EntryNo := EntryNo + ExprAr [NoExpr]. Expr. EntryNo
               End ; { If & For }

             SimplifyExpression (IMatrixRow [NoPar])

           End { With & If & For & With }
   End { With }
 End ;

 { *** list a model *** }
```

```
Procedure ListModel (Var Model : ModelType ;
                         Index : Integer    ) ;
Var
  NoEqn : EquationRange  ;
  NoPar : ParameterRange ;
Begin
  With Model, Equations, Parameters Do
  Begin
    Writeln ;

    Write ('Model no   : ', Index, LF,
           'Model name : ', Name , LF,
           'Equations', LF,
           '---------', LF ) ;

    For NoEqn:=1 To EqnNo Do
      Writeln (NoEqn :2, '. ', EqnInfo [NoEqn]. Infix) ;

    Write ('Parameters (Name & Type)', LF,
           '----------', LF) ;

    For NoPar:=1 To ParNo Do
    Begin
      Write  (NoPar :2, '. ', ParInfo [NoPar]. Name, '  ') ;
      Case ParInfo [NoPar]. Type_ Of
        Exo  : Write ('Exo  var') ;
        Endo : Write ('Endo var') ;
        Cnst : Write ('Constant')
      End ;
      WriteSc ('')
    End ;

    Writeln
  End { With }
End ;

{ *** list models in the library *** }
Procedure ListLibrary ;
Var
  Model : ModelTypePtr ;
Begin
  Reset (ModelLibFile) ;

  Model := Ptr (HeapPtrSeg, HeapPtrOfs) ;

  While Not Eof (ModelLibFile) Do
  Begin
    Read (ModelLibFile, Model^) ;

    ListModel (Model^, FilePos (ModelLibFile)) ;

    Writeln
  End ;

  Message (5)
End ;

{ *** delete a model from library *** }
Procedure DeleteFromLibrary ;
Var
  TempFile : ModelLibraryFileType ;
  Model    : ModelTypePtr         ;
  Name     : ModelNameType        ;
  Index    : Integer              ;
Begin
  Model := Ptr (HeapPtrSeg, HeapPtrOfs) ;

  Write (LF, 'Model name : ') ;
  ReadInputS (ModelNameLen, Name) ;
  If Not ModelExists (Name, Index) Then
    GiveError (18, 0) ;

  Assign  (TempFile, 'TEMP') ;
  Rewrite (TempFile) ;

  Reset (ModelLibFile) ;
  While Not Eof (ModelLibFile) Do
  Begin
    Read (ModelLibFile, Model^) ;
    If FilePos (ModelLibFile) - 1 <> Index Then
      Write (TempFile, Model^)
  End ;

  Close  (ModelLibFile) ;
  Close  (TempFile) ;

  Erase  (ModelLibFile) ;
  Rename (TempFile, ModelLibraryFileName) ;

  Assign (ModelLibFile, ModelLibraryFileName) ;

  Message (6)
End ;

{ *** insert current model into library *** }
Procedure SaveCurrentModel ;
Var
  Index : Integer ;
Begin
  If ModelExists (CrModel. Model. Name, Index) Then
    GiveError (19, 0) ;

  Seek  (ModelLibFile, FileSize (ModelLibFile)) ;
  Write (ModelLibFile, CrModel. Model) ;

  Message (4)
End ;

{ *** input a model *** }
Procedure ReadModel (Var Model : ModelType) ;
Type
  ParInfoType = Array [ParameterRange] Of Record
                                            Name_  : ParameterName  ;
                                            Index_ : ParameterRange ;
                                            Type_  : 0 .. 3
                                          End ;
Var
  SubModel  : ModelTypePtr   ;
  Name      : ModelNameType  ;
  Index     : Integer        ;
  Info      : ParInfoType    ;
  EqnStr    : StringType     ;
  NoEqn     : EquationRange  ;
  NoPar ,
  i         : ParameterRange ;
  StrPos ,
  NamePos   : Integer        ;
  TypeCh    : Char           ;

  Procedure ChangeParIndex (Var NewExpression : PostfixType   ;
                                OldExpression : PostfixType   ;
                                OldIndex ,
                                NewIndex      : ParameterRange ) ;
  Var
    NoEntry : PostfixEntryRange ;
  Begin
    With OldExpression Do
```

```
      For NoEntry:=1 To EntryNo Do
        With Entry [NoEntry] Do
          If (Type_ = Parameter_) And (Index = OldIndex) Then
            NewExpression. Entry [NoEntry]. Index := NewIndex
  End ;

Begin
  SubModel := Ptr (HeapPtrSeg, HeapPtrOfs + $7FFF) ;

  With Model, Equations, Parameters Do
  Begin
    Write ('Model name : ') ;
    ReadInputS (ModelNameLen, Name) ;

    If ModelExists (Name, Index) Then
      GiveError (19, 0) ;

    EqnNo := 0 ;
    ParNo := 0 ;

    For NoEqn:=1 To MaxEquation Do
      For NoPar:=1 To MaxParameter Do
        NumberEntry (EqnInfo [NoEqn]. IMatrixRow [NoPar], 0) ;

    For NoPar:=1 To MaxParameter Do
      Info [NoPar]. Type_ := 0 ;

    Writeln (LF, 'Equations',
             LF, '---------') ;

    Repeat
      Write ('Eqn : ') ;

      ReadInputS (70, EqnStr) ;

      If TrimL (EqnStr) <> '' Then
        If EqnStr [1] = '@' Then
        Begin
          If Not ModelExists (Copy (EqnStr, 2, Length (EqnStr) - 1), Index) Then
            GiveError (18, 0) ;

          Seek (ModelLibFile, Index) ;
          Read (ModelLibFile, SubModel^) ;

          ListModel (SubModel^, Index+1) ;

          If EqnNo + SubModel^. Equations. EqnNo > MaxEquation Then
            GiveError (21, 0) ;

          For NoPar:=1 To SubModel^. Parameters. ParNo Do
          Begin
            Write (SubModel^. Parameters. ParInfo [NoPar]. Name, ' --> ') ;
            ReadInputS (ParameterNameLen, Info [NoPar]. Name_) ;
            If TrimL (Info [NoPar]. Name_) = '' Then
              Info [NoPar]. Name_ := SubModel^. Parameters. ParInfo [NoPar]. Name ;
            If Not ParameterExists (Parameters, Info [NoPar]. Name_, Info [NoPar]. Index_) Then
            Begin
              If ParNo = MaxParameter Then
                GiveError (22, 0) ;
              ParNo := ParNo + 1 ;
              ParInfo [ParNo]. Name  := Info [NoPar]. Name_ ;
              Write ('Type (0:Default,1:Endo,2:Exo,3:Constant) : ') ;
              ReadChar (TypeCh, ['0'..'3']) ;
              If TypeCh = '0'
                Then Info [ParNo]. Type_ := Ord (SubModel^. Parameters. ParInfo [NoPar]. Type_) + 1
                Else Info [ParNo]. Type_ := Ord (TypeCh) - 48
            End ;
            SubModel^. Parameters. ParInfo[NoPar]. Name := TrimT (SubModel^.Parameters.ParInfo[NoPar].Name)
          End ; {For }
        Writeln ;

        For NoEqn:=EqnNo+1 To SubModel^. Equations. EqnNo + EqnNo Do
          With EqnInfo [NoEqn] Do
          Begin
            Infix      := SubModel^. Equations. EqnInfo [NoEqn-EqnNo]. Infix      ;
            Expression := SubModel^. Equations. EqnInfo [NoEqn-EqnNo]. Expression ;
            For NoPar:=1 To SubModel^. Parameters. ParNo Do
            Begin
              StrPos := 1 ;
              With SubModel^. Parameters. ParInfo [NoPar], Info [NoPar] Do
                Repeat
                  NamePos := Pos (Name, Copy (Infix, StrPos, 255)) ;
                  If (NamePos <> 0) And
                     (Copy (Infix, StrPos+NamePos-1, Length (Name)) = Name) And
                     ((Not (Infix [StrPos+NamePos-1+Length(Name)] IN ['A'..'Z','0'..'9'])) Or
                      (StrPos+NamePos-1+Length(Name) > Length (Infix)))
                  Then Begin
                    Infix := Copy (Infix,1,StrPos+NamePos-2)+ TrimT(Name_)+
                             Copy(Infix,StrPos+NamePos-1+Length(Name),255) ;
                    StrPos := StrPos + NamePos - 1 + Length (TrimT(Name_))
                  End
                  Else
                    StrPos := StrPos + NamePos
                Until NamePos = 0 ;
              ChangeParIndex (Expression, SubModel^. Equations. EqnInfo [NoEqn-EqnNo]. Expression,
                              NoPar, Info [NoPar]. Index_) ;
              IMatrixRow[Info[NoPar].Index_]:=SubModel^.Equations.EqnInfo[NoEqn-EqnNo].IMatrixRow[NoPar];
              For i:=1 To SubModel^. Parameters. ParNo Do
                ChangeParIndex (IMatrixRow [Info [NoPar]. Index_],
                                SubModel^. Equations. EqnInfo [NoEqn-EqnNo]. IMatrixRow [NoPar],
                                i, Info [i]. Index_)
            End { For }
          End ; { With & For }

          EqnNo := EqnNo + SubModel^. Equations. EqnNo
        End { If ... Then Begin }
        Else Begin
          EqnNo := EqnNo + 1 ;
          EqnInfo [EqnNo]. Infix := EqnStr ;
          ParseEquation (Model, EqnNo, EqnStr) ;
          DerivativeOfEquation (Model, EqnNo)
        End
    Until (TrimL (EqnStr) = '') Or (EqnNo = MaxEquation) ;

    If ParNo = 0 Then
      GiveError (23, 0) ;

    Write ('Parameter types (1:Endo  2:Exo  3:Constant)', LF,
           '----------------', LF) ;

    For NoPar:=1 To ParNo Do
    Begin
      If (Info [NoPar]. Type_ = 0) Then
      Begin
        Write (ParInfo [NoPar]. Name, ' : ') ;
        ReadChar (TypeCh, ['1'..'3']) ;
        Info [NoPar]. Type_ := Ord (TypeCh) - 48
      End ;
      ParInfo [NoPar]. Type_ := TypeOfParameter (Info [NoPar]. Type_ - 1)
    End
  End { With }
End ;

{ *** insert a model into library *** }
Procedure InsertToLibrary ;
Var
  Model : ModelTypePtr ;
```

```
Begin
  Model := Ptr (HeapPtrSeg, HeapPtrOfs) ;

  ReadModel (Model^) ;

  Seek  (ModelLibFile, FileSize (ModelLibFile)) ;
  Write (ModelLibFile, Model^) ;
  Message (7)
End ;

{ *** initialize info of current model *** }
Procedure InitCrModelInfo ;
Var
  NoEqn  : EquationRange  ;
  NoPar  : ParameterRange ;
Begin
  With CrModel, ModelInfo, Model, Equations, Parameters Do
  Begin
    EndoParNo := 0 ;
    ExoParNo  := 0 ;

    For NoPar:=1 To ParNo Do
      With ParInfo [NoPar] Do
        If Type_ = Endo Then
        Begin
          EndoParNo := EndoParNo + 1 ;
          EndoParAr [EndoParNo] := NoPar
        End
        Else Begin
          ExoParNo := ExoParNo + 1 ;
          ExoParAr [ExoParNo] := NoPar
        End ;

    If EqnNo <> EndoParNo Then
      GiveError (7, 0)
  End ; { With }

  CrModelGiven   := True  ;
  IMatrixSGiven  := False ;
  ParValuesGiven := False
End ;

{ *** load a model from library *** }
Procedure LoadCurrentModel ;
Var
  Name  : ModelNameType ;
  Index : Integer       ;
Begin
  Write (LF, 'Model name : ') ;
  ReadInputS (ModelNameLen, Name) ;

  If Not ModelExists (Name, Index) Then
    GiveError (18, 0) ;

  CrModelGiven := False ;

  Seek (ModelLibFile, Index) ;
  Read (ModelLibFile, CrModel. Model) ;

  InitCrModelInfo ;

  Message (8)
End ;

{ *** read a current model *** }
Procedure ReadCurrentModel ;
Begin
  CrModelGiven := False ;
  ReadModel (CrModel. Model) ;

  InitCrModelInfo ;

  Message (9)
End ;



{
*** Turbo Pascal routines used in the program that are not available in Standard Pascal ***
  Procedures :
    ClrScr  --> clears screen, places cursor in the upper left-hand corner.
    GotoXY (Column, Row) --> moves cursor to the specified location.
    FillChar (Var, Num, Value) --> fills Num bytes of memory, starting at
             the first byte of Var, with Value.
  Functions:
    Abs (Num)    --> absolute value of Num.
    ArcTan (Num) --> angle, in radians, whose tangent is Num.
    Cos (Num)    --> cosine of Num which is in radians.
    Sin (Num)    --> sine of Num which is in radians.
    Exp (Num)    --> exponential of Num.
    Int (Num)    --> integer part of Num.
    Ln (Num)     --> natural logarithm of Num.
    Sqr (Num)    --> square of Num.
    Sqrt (Num)   --> square root of Num.
    Odd (Num)    --> true if Num is an odd number.
    Ord (Var)    --> ordinal number of Var in the set defined by the type Var.
    Round (Num)  --> the value of Num rounded.
    KeyPressed   --> true if a key is pressed at the console.
    UpCase (Ch)  --> uppercase of Ch.
}
```

BIBLIOGRAPHY

1. Luenberger, D.G., Introduction to Dynamic Systems: Theory, Models, and Applications. New York: Wiley, 1979.

2. De Kleer, J. and J.S. Brown, "A Qualitative Physics Based on Confluences," Artificial Intelligence, Vol.24, pp.7-83, 1984.

3. De Kleer, J. and J.S. Brown, "Theories of Causal Ordering," Artificial Intelligence, Vol.29, pp.33-61, 1986.

4. Iwasaki, Y. and H.A. Simon, "Causality in Device Behavior," Artificial Intelligence, Vol.29, pp.3-32, 1986.

5. Kuipers, B., "Qualitative Simulation," Artificial Intelligence, Vol.29, pp.289-338, 1986.

6. Kuipers, B., "Commonsense Reasoning About Causality: Deriving Behavior from Structure," Artificial Intelligence, Vol. 24, pp.169-203, 1984.

7. Weld, D.S., "Comparative Analysis," Artificial Intelligence, Vol.36, pp.333-373, 1988.

8. Forbus, K.D., "Qualitative Process Theory," Artificial Intelligence, Vol.24, pp.85-168, 1984.

9. Weld, D.S., "The Use of Aggregation in Causal Simulation," Artificial Intelligence, Vol.30, pp.1-34, 1986.

10. Hayes, P.J., "The Naive Physics Manifesto," D. Michie (Ed.), Expert Systems in the Microelectronic Age (Edinburgh University Press, Edinburgh, 1979).

11. Hayes, P.J., "The Second Naive Physics Manifesto," J. Hobbs and R. Moore (Eds.), Formal Theories of the Commonsense World (Ablex, Norwood, NJ, 1985).

12. Williams, B., "Qualitative Analysis of MOS Circuits," Artificial Intelligence, Vol.24, pp.281-346, 1984.

13. Gandolfo, G., Mathematical Methods and Models in Economic Dynamics. Amsterdam, North-Holland Pub. Co., 1971.

14. Franks, R.G.E., Mathematical Modeling in Chemical Engineering. John Wiley & Sons Inc., 1967.

15. Lapp, S.A. and G.J. Powers, "Computer-aided Synthesis of Fault Trees," IEEE Transactions on Reliability, pp.2-13, April 1977.

16. Moore, R.E., Interval Analysis. Englewood Cliffs, N.J., Prentice-Hall, Inc., 1966.

17. Dahlquist, G. and Å. Björck, (Tr. by N. Anderson), Numerical Methods. Prentice-Hall, Inc., 1974.