CARNEGIE MELLON UNIVERSITY

REPRESENTATION, ANALYSIS AND SOLUTION OF CONDITIONAL MODELS IN AN EQUATION-BASED ENVIRONMENT

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree of

DOCTOR OF PHILOSOPHY

in

CHEMICAL ENGINEERING

by

VICENTE RICO-RAMIREZ

Pittsburgh, Pennsylvania

August, 1998

TO MOM AND DAD

TO ROSSY

IN MEMORY OF EDITH

Abstract

Process modeling is an important task in many process engineering activities. At the lowest level, process models are represented by a large set of variables and a large system of linear and nonlinear equations that relate them. The equation-based modeling approach has been demonstrated as effective in solving simulation, optimization, parameter estimation and data reconciliation problems. Even though many currently available equation-based modeling systems have been reported in the literature, most of them give little or no attention to conditional models.

Conditional models exist when the equations defining a system depend on where the model solution lies. Examples of conditional models in chemical engineering are systems involving physicochemical discontinuities such as flow and phase transitions. This work investigates the setting up and solving of conditional models within an equation-based modeling environment.

We first describe modeling tools for the efficient representation of conditional models and give the details of their computer implementation. We present examples to demonstrate the scope of application of the new tools.

Then, we discuss an extension of a previous approach to the consistency analysis of conditional models which aids in the proper selection of the degrees of freedom for such models. We also show how, by taking advantage of the structure of the problem, it is often possible to reduce the effort required by the proposed consistency analysis.

Finally, we focus on the implementation and testing of some approaches to solving conditional models. Specifically, we describe an implementation of a boundary crossing algorithm in an equation-based environment, and an extension of the standard complementarity formulation for the representation of conditional models. We also explore numerical techniques for solving the proposed complementarity representation of a conditional model and solve several examples demonstrating the advantages and disadvantages of each of the approaches to solving.

Acknowledgments

I have to first thank my research advisor, Professor Arthur Westerberg, for his help and guidance over the past four years. My appreciation goes to Professor Edmond Ko for giving me an opportunity to pursue graduate studies at Carnegie Mellon University. Many thanks also go to “Consejo Nacional de Ciencia y Tecnología (CONACYT)” for funding most of this research project.

To members of the ASCEND group, Ben Allan, Mark Thomas, Ken Tyner, Chad Farschman and Duncan Coffey for their assistance and collaboration over the years.

To my friends Octavio, Antonio, Héctor, Arturo, Juan, Weimin, Jason and Margaret for their moral support and encouragement. To members of the “sudsuckers” soccer team for making my time in Pittsburgh sometimes very enjoyable.

I would specially like to thank Salvador, Misael, José Luis, Agustín, Enrique, Silvestre, Hugo and Yalí for the roles they have played in my life.

Of course, I must thank my parents, Esther Ramírez and José Inés Rico, and my brothers and sisters for putting up with me through this difficult period of my life. Finally, I want to thank Rossy, whom never lost faith nor love despite the years and the distance.

Table of Contents

List of Figures

List of Tables

CHAPTER 1 INTRODUCTION

Chapter 1 provides an introduction to this research. A brief review of the state of the art in process modeling is presented. Also, the main concepts involved in the area of conditional modeling are discussed. Finally, the motivation, the goal and the outline of the thesis are given.

1.1 EQUATION-BASED MODELING

Modeling is the process of mapping reality into a representation that is thought to be useful for understanding that reality (Abbott, 1996). Process modeling is an important task in many process engineering activities. The state of the art as well as future trends in process modeling and simulation have been reviewed in a number of publications (Piela, 1989; Boston *et al*., 1993; Pantelides and Britt, 1994; Marquardt, 1996).

The modeling tools in current simulators may roughly be classified into two groups: modular oriented and equation-based approaches (Boston *et al.*, 1993). A detailed discussion of the modular approach versus the equation-based approach can be found in Marquardt (1996).

On the one hand, modular approaches address modeling on the flowsheet level. Every process is abstracted by a block diagram consisting of standardized blocks which model the behavior of a process unit or a part of it. All the blocks are linked by connections representing the flow of information, material and energy employing standardized interface and stream formats (Marquardt, 1996). This modular approach, though powerful and easily accessible to many engineers for the solution of standard flowsheet problems, does not adequately support the solution of more involved problems. This is due to the lack of precoded models for many unit operations of adequate level of detail (Marquardt, 1996).

On the other hand, equation-based modeling tools are motivated by the fact that, at the lowest level, process models are represented by a large set of variables and a large system of linear and nonlinear equations that relate them. Thus, equation-based approaches support the implementation of unit models by means of declarative modeling languages. In the equation-based modeling approach, the definition of the system of equations is independent of any particular application or solution algorithm that may be used for their solution. For that reason, the solution to equation-based models has been demonstrated as effective in solving simulation, optimization, parameter estimation and data reconciliation problems, all using a single set of equations (Allan, 1997). Recognition of the potential benefits of the equation-based technology has led to the development of equation-based modeling tools such as SpeedUP (Pantelides, 1988), gPROMS (Barton, 1992; Oh and Pantelides, 1994), and GAMS (Brooke *et al*., 1997). Some other researches have proposed that equation-based modeling can be further facilitated by the use of object-oriented frameworks. Examples are OMOLA (Mattsson and Andersson, 1993) and ASCEND (Piela, 1989; Allan, 1997).

Arguably, the state of the art in process modeling software at the time of writing this work is an object-oriented, equation-based modeling environment (Abbott, 1996; Allan, 1997).

1.2 CONDITIONAL MODELS

Process engineering design and simulation require one to find solutions to a large system of nonlinear equations. Conventional process models consist of a set of variables and a unique set of *m* equations that related them. On the other hand, conditional models constitute a means to formulate alternative sets of *m* equations depending on the values of the modeling variables. Thus, in conditional models the system of equations of the model is different for each of the alternatives. This work is particularly concerned with the issues involved in the modeling and solution of conditional models.

A conditional model, as defined by Grossmann and Turkay (1996), consists of a system of equations expressed by two sets: a globally defined invariant set of equations and a variant (or locally defined) set of conditional equations which are expressed as disjunctions. Grossmann and Turkay (1996) show that a conditional model can be represented as the system of disjunctive equations:

$$\begin{aligned} &\underline{h}(\underline{x}) = 0 \\ &\underset{i \in D_k}{\vee} \begin{bmatrix} r_{i_{jk}}(\underline{x}) = 0 \\ g_{i_{lk}}(\underline{x}) \le 0 \end{bmatrix} k \in K \qquad \begin{matrix} \forall j \in [1 \ldots \beta_k] \\ \forall l \in [1 \ldots \gamma_k] \end{matrix} \\ &\underline{x} \in R^n \end{aligned} \qquad \textbf{(1.1)}$$

where $\underline{h}(\underline{x})$ and $\underline{r}_{i_k}(\underline{x})$ represent the vectors of the invariant and the variant sets of equations respectively, K represents the set of disjunctions and the index i is used to indicate the i-th disjunctive term in each disjunction D_k. The vector $\underline{h}(\underline{x})$ is m-dimensional, and it is assumed that $\underline{r}_{i_k}(\underline{x})$ is β_k-dimensional and that $\underline{g}_{i_k}(\underline{x})$ is γ_k-dimensional, for all i in D_k. The equations $\underline{h}(\underline{x})$ can be said to be defined over the entire feasible region, while the inequalities $\underline{g}_{i_k}(\underline{x})$ define the domain of validity of each variant set of equations $\underline{r}_{i_k}(\underline{x})$. In that way, each variant set of equations is confined to some subregion resulting from the dissection of the feasible region (Zaher, 1991). The solution to the system will be given by the vector $\hat{\underline{x}}$ satisfying the invariant set $\underline{h}(\underline{x})$ and exactly one set of equations for each of the disjunctions, providing that the corresponding set of inequalities is satisfied. Grossmann and Turkay (1996) also address the existence and uniqueness of the solution for the linear case.

Examples of conditional models in chemical engineering are systems involving physicochemical discontinuities such as flow and phase transitions. There are also cases where conditional cost functions are used in the optimization of process flowsheets. Figure 1-1, taken from the work of Zaher (1995), illustrates the case of a flow transition. In that case, we must incorporate alternative equations for the transport properties and velocity bounds within the model, using the simultaneously calculated value of the flow type indicator (Reynolds or Mach number) relative to some critical value to determine which is active. A simple flash equilibrium calculation also represents a conditional model commonly encountered in chemical engineering. Figure 1-2 illustrates that example. Depending on the value of the temperature (relative to the dew point temperature and bubble point temperature at a given pressure), the system may be present as a subcooled liquid, as a superheated vapor, or as a liquid-vapor mixture in equilibrium. The system of equations of the model is different for each of the alternatives. The problem is that the temperature may be unknown, and, therefore, there is no way of knowing *a priori* which set of equations one needs to satisfy.

According to the above, the main difficulty while dealing with conditional models is that their solution involves simultaneously selecting the equations to be solved and solving them.

Figure 1-1 Fluid flow transition.

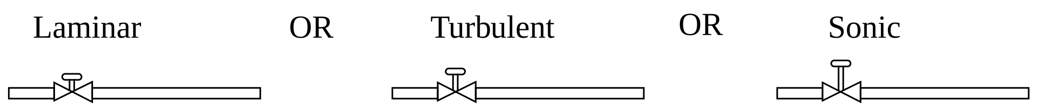
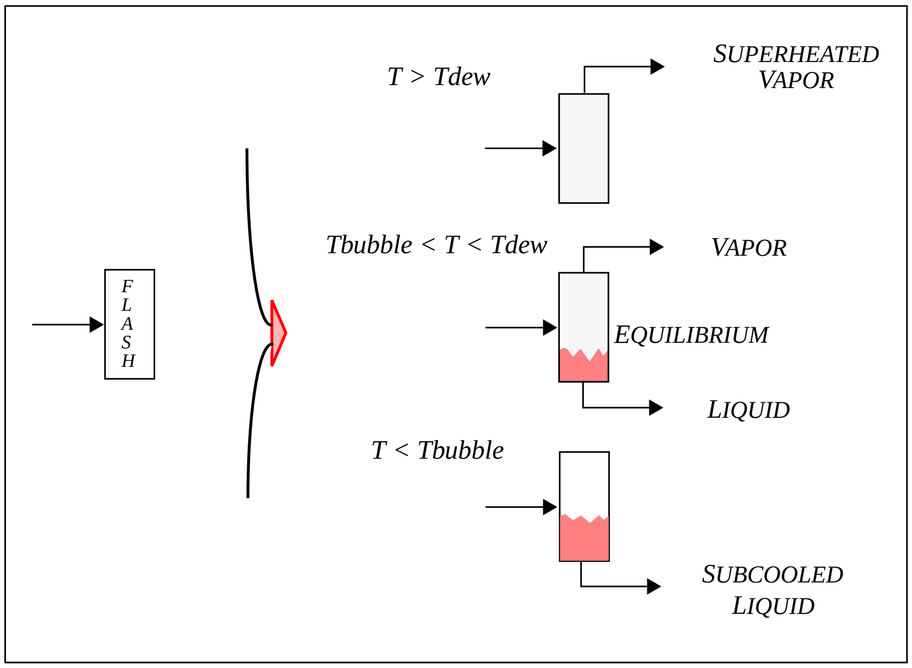


Figure 1-2 A conditional model: flash calculation.



1.3 MOTIVATION: CONDITIONAL MODELING IN EXISTING MODELING ENVIRONMENTS

Many currently available modeling systems have been reported in the literature. Some of the most recent approaches to computer-aided modeling tools are shown in Table 1-1.

Table 1-1 Recent modeling systems

System	Reference
ABACUSS	Feehery and Barton (1996)
ASCEND	Piela (1989)
DYMOLA	Elmqvist *et al*. (1993)
GAMS	Brooke *et al*. (1997)
gPROMS	Barton (1992)
MODEL.LA	Stephanopoulos *et al*. (1990)
SASE	Garrett and Hakim (1992)
SPL/SDL	Kiliccote (1996)
SpeedUp	Pantelides (1988)
OMOLA	Mattson and Anderson (1993)
VEDA	Bogusch and Marquardt (1995)
VERILOG	Thomas and Moorby (1996)

On the one hand, there is a group of modeling languages mainly suitable for their use in a specific application domain. In MODEL.LA and VEDA, elements tailored to chemical engineering applications are included in the language definition; VERILOG is a hardware description language used to design and test electronic systems; SASE and SPL/SDL are languages to represent the standards imposed on the design of civil engineering structures and facilities. In such languages, very efficient modeling constructs exist, but their main limitation is that they have been developed to match the issues of specific engineering applications.

On the other hand, there is a group of general modeling languages: ABACUSS, ASCEND, DYMOLA, GAMS, gPROMS, SpeedUp and OMOLA. In such languages, elements are not restricted to specific engineering applications.

All of the general equation-based modeling languages presented in Table 1-1 provide constructs which allow the representation of conditional dependence of some of the model equations. For instance, the IF-THEN-ELSE construct (ABACUSS, ASCEND, gPROMS, DYMOLA, SpeedUp and OMOLA), the CASE construct (gPROMS) and the WHEN construct (DYMOLA). Two different applications of these modeling constructs have been developed:

1. First, the conditional language constructs have been used to handle changes procedurally of the configuration of the problem because of time or state events occurring during the dynamic simulation of a system, assuming that the initial steady state of the system is known a priori.
2. Secondly, some of the equation-based approaches, such as gPROMS and SpeedUp, have acknowledged the difficulty of solving steady state problems where the clause of an IF-THEN-ELSE construct which is active cannot be specified explicitly, but it has to be automatically calculated (Barton, 1992). It is not clear, however, how these modeling environments handle this problem, since an explicit methodology is not provided.

Here we have to make an important distinction about these two applications of the existing conditional language constructs:

- In conditional models, as defined by Equation (1.1), the conditional dependence of the model equations does not necessarily exist because of state or time events occurring during the dynamic simulation of a system. Hence, the problem of discrete event dynamic simulation described first is not the main focus of attention of this research.
- On the contrary, the problem described as the second application of the existing conditional language constructs fits perfectly into our definition of a conditional model given by (1.1).

Accordingly, we can argue that most of the existing equation-based modeling languages have not presented a formal approach to solving conditional models as formulated by Equation (1.1). We should note though that the GAMS modeling language provides modeling and solving tools (such as the dollar condition for conditional equations and the if-else statement for flow control) powerful enough to represent and solve conditional models. The major disadvantage of the GAMS modeling language is that it does not provide an object-oriented framework to facilitate the modeling process.

This limited attention from the equation-based community to conditional models provides the main motivation for this research project. Several problems arise when building and testing models of complex processes in an equation-based environment: *i)* providing initial guesses, *ii)* analyzing degrees of freedom, and *iii)* efficiently computing solutions. The nature of a conditional model makes the solution of these problems an even more difficult task.

1.4 OUR GOAL

The goal of this work is to investigate the setting up and solving of conditional models within an equation-based modeling environment. It is important to distinguish between two separated (but complementary) activities of this research project:

1. Efficient representation of a conditional model in an equation-based modeling environment. Following the equation-based approach, this representation is intended to be independent of any particular application, or of any solver or algorithm used for finding a solution to the system of equations. Potential applications of this capability vary from the simple substitution of one equation for another (as in the case of the laminar-turbulent flow transition), to the substitution of a section of a chemical plant for another (as can be required while analyzing and initializing a superstructure).
2. Implementation and testing of alternative approaches to the solution of a conditional model. It is important to recognize that modeling is a user dependent task. Frequently, different modelers produce different formulations for the same problem and, as a consequence, different approaches and techniques may be used to find a solution to it. One can formulate conditional models as mixed-integer programming problems. See for example Grossmann and Turkay (1996). Besides the mixed-integer formulation, some alternative approaches exist. Research in this work will focus on

two of them:

- a boundary crossing algorithm (Zaher, 1995) and
- a complementarity representation of conditional models.

In particular, we will propose an approach for the implementation of the boundary crossing algorithm in an equation-based environment, and an extension of the standard complementarity formulation for the representation of conditional models. Then, we will explore some alternative numerical techniques for solving the proposed complementarity representation of a conditional model.

1.5 OUTLINE

Given the background and the goal of this work, the rest of this thesis is organized as follows:

In Chapter 2, we describe modeling tools for an efficient representation of conditional models and give the details of their computer implementation. Also, we use several chemical engineering examples to demonstrate the scope of application of these new modeling capabilities.

In Chapter 3, we present a brief review on the topic of structural analysis of conditional models. Then, we provide an extension to the consistency analysis of conditional models developed by Zaher (1993). This extension allows the consistency analysis to be applied to conditional models in which the number of variables and equations for each of the alternatives is not the same. Also, we show how, by taking advantage of the structure of the problem, it is sometimes possible to reduce the combinatorial effort required by such an analysis.

In Chapter 4, we investigate the solving of conditional models using the boundary crossing algorithm. We give the details of our practical implementation of this technique and solve several examples of conditional models in chemical engineering. Finally we discuss the scope and limitations of the algorithm.

In Chapter 5, we provide a complementarity formulation for representing algebraic systems of disjunctive equations. We show that this formulation not only represents an alternative to MINLP formulations avoiding discrete decisions, but also avoids the need for special procedural nonlinear techniques as required by the boundary crossing algorithm. We identify the advantages and disadvantages associated with the complementarity formulation and study its solution by using a conventional nonlinear solver and pivotal techniques.

In Chapter 6, we investigate the solving of the complementarity formulation described in Chapter 5 by using interior point methods. Following a brief review of the fundamentals of interior point methods, we describe the globally convergent framework proposed by Wang et al (1996) for solving a constrained system of nonlinear equations by an interior point potential reduction method. Then, we show how we can apply the algorithm proposed by Wang to solve the complementarity examples used as cases of study throughout this work. After that, we modify Wang's algorithm by applying some high order strategies designed to improve convergence (Mehrotra, 1992; Gondzio, 1996), and compare the results obtained with each of the methods.

Finally, In Chapter 7 we conclude the thesis by giving a summary of the work and contributions made in this research. Also directions and recommendations for future work are highlighted.

1.6 REFERENCES

Abbott, K. A.; Very Large Scale Modeling. Ph. D. Thesis. Department of Chemical Engineering Carnegie Mellon University, Pittsburgh, PA, 1996.

Allan, B. A.; A More Reusable Modeling System; Ph.D. thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA,1997.

Barton, P. I.; The Modeling and Simulation of Combined Discrete/Continuous Processes. Ph.D. thesis, Department of Chemical Engineering, Imperial College of Science, Technology and Medicine, 1992.

Bogusch, R. and Marquardt, W.; A Formal Representation of Process Model Equations. *Comput. Chem. Eng.*, 19:S211-216, 1995.

Boston, J. F., Britt, H. I. and Tayyabkhan, M. T.; Software: Tackling Tougher Tasks. *Chem. Eng. Progr*. Nov., 38-49, 1996.

Brooke, A., Kendrick, D., Meeraus, A. and Raman, R.; GAMS - Language Guide. GAMS Development Corporation, 1997.

Elmqvist, H., Cellier, F.E. and Otter, M.; Object-Oriented Modeling of Hybrid Systems. ESS'93 European Simulation Symp., Delft, The Netherlands, 25-28, 1993.

Feehery, W. F. and Barton, P. I.; A Differentiation-Based Approach to Dynamic Simulation and Optimization with High Index Differential-Algebraic Equations. *Computational Differentiation*, M. Berz, C. Bischof, G.Corliss, and A. Griewank Editors, SIAM, 1996.

Garrett, J. H. and Hakim, M. M.; Object-Oriented Model of Engineering Design Standards. *Journal of Computing and Civil Engineering*, 6, 323-347, 1992.

Grossmann, I. E. and Turkay, M.; Solution of Algebraic Systems of Disjunctive Equations. *Comput. Chem. Eng.*, 20:S339–44, 1996. Suppl. Part A.

Kiliccote, H.; A Standards Processing Framework. PhD thesis, Department of Civil Engineering, Carnegie Mellon University, Pittsburgh, PA, 1996.

Marquardt, W.; Trends in Computer-Aided Process Modeling. *Comput. Chem. Eng.*, 20(6):591-609, 1996.

Mattsson, S, E., Andersson, M.; OMOLA - An Object Oriented Modeling Language. *Recent Advances in Computer Aided Control Engineering*. Elsevier, 291-310, Amsterdam, 1993.

Oh, M. and Pantelides, C. C.; A Modeling and Simulation Language for Combined Lumped and Distributed Parameter Systems. *5th Int. Conf. PSE'94*, 1, 37-44, Kyongju, Korea, 1994.

Pantelides, C. C.; SPEEDUP-Recent Advances in Process Simulation. *Comput. Chem. Eng.*, 12(7):745–755, 1988.

Pantelides, C. C., Britt, H. I.; Multipurpose Process Modeling Environments. *FOCAPD'94*, Snowmass, CO., 1994.

Piela, P.; ASCEND: An Object-Oriented Computer Environment for Modeling and Analysis. PhD thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, 1989.

Stephanopoulos, G., Henning, H., and Leone, H.; MODEL.LA. A Language for Process Engineering, Parts I and II. *Comput. Chem. Eng.*, 14, 813-869, 1990.

Thomas, D. E. and Moorby, P. R.; The Verilog Hardware Description Language, Third Edition, Kluwer Academic Press, Boston, 1996.

Zaher, J. J.; Conditional Modeling. Ph. D. thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, 1995.

Zaher, J. J.; Conditional Modeling in an Equation-Based Modeling Environment. The Annual AIChE National Meeting, 1991. paper 138c.

Zaher, J. J.; Conditional Programming. The Annual AIChE National Meeting, March 1993.

CHAPTER 2 CONDITIONAL MODELING TOOLS

In this chapter we identify the modeling capabilities needed for an efficient representation of conditional models: conditional configuration of a model structure, conditional compilation and conditional execution of procedural statements. We then describe modeling tools for the performance of each of the identified tasks. We next describe the details of the computer implementation of these tools and show how the expressiveness of an equation-based modeling language increases with their incorporation. Finally we present several chemical engineering examples to demonstrate the scope of application of the proposed extensions.

2.1 BACKGROUND

Previous implementations of conditional statements in an equation-based modeling environment have been reported. One such mechanism is the IF-THEN-ELSE construct of *SpeedUp* described by Pantelides (1988):

```
IF logical_condition THEN
     equation1
ELSE
     equation2
ENDIF
```

where both the logical conditions and the equations are expressed in terms of the model variables. Such a construct defines two system states corresponding to the IF and the ELSE clauses respectively. Multiple states may be described by nesting several IF statements. Other implementations of the IF-THEN-ELSE construct have also been reported by Barton (1992) with the IF-Equation of *gPROMS*, and by Feehery and Barton (1996) with the IF structure of ABACUSS.

Barton (1992) and Barton and Pantelides (1994) also incorporated the CASE equation into *gPROMS*. The CASE equation is used to define both the appropriate modeling equations in each state and the logical conditions for transitions among states. It covers multiple states within only one statement and has the advantage of successfully representing irreversible discontinuities.

There is a major difficulty in the previous approaches as tools for conditional modeling. They only allow the substitution of a list of primitive equations (or arrays of equations) for another. In an equation-based modeling environment in which object oriented concepts like hierarchy (building complex models from small models) and inheritance are constantly in use, this approach places a significant limitation on one's modeling efficiency. For instance, it makes it very difficult to model unit replacement when searching over a superstructure.

Only for the purpose of model representation, in this chapter we extend the scope of the definition of a conditional model. Here, we consider as a conditional model any problem including a set of disjunctive statements as part of its formulation. In other words, the domain of validity of each particular alternative set of equations does not have to be given by inequality constraints. Any kind of logical, integer or binary variables can be used for that purpose instead. With that in mind, the expressiveness of our modeling tools can also be applied to problems on which the selection of alternative configurations is based on logic, algorithmic and heuristic decisions.

2.2 CONDITIONAL MODELING TOOLS

Equation oriented modeling tools support the implementation of unit models and their incorporation in a model library by means of a declarative modeling language; declarative in the sense of explicit and symbolic encapsulation of the knowledge about models (Marquardt, 1996). Moreover, methods must also be attached to a model definition for the numerical processing of the model equations.

In this chapter, we identify three modeling capabilities which support the efficient development of conditional models in both the declarative definition of equation-based models and the procedural execution of methods:

- Conditional configuration of the model structure.
- Conditional compilation.
- Conditional execution of the procedural code of methods.

In the remainder of this section, we describe the syntax and semantics of modeling tools which allow the practical implementation of those modeling capabilities.

2.2.1 CONDITIONAL CONFIGURATION OF A MODEL STRUCTURE

Several numerical algorithms and methodologies for the solution of conditional models have recently become available. See for example Zaher (1995), Grossmann and Turkay (1996) and Turkay and Grossmann (1996). A common characteristic of any of these methodologies is that the dynamic switching among alternative model configurations is required during the solution process. Hence, a declarative modeling language has to provide a means to represent all the alternative structural configurations of the problem as well as the conditions which trigger the switching among them. Next, we describe language constructs which fulfill this requirement:

- the WHEN statement, which provides an efficient means to declare alternative modeling configurations
- logical relations and the CONDITION statement, which can be used to represent the conditions triggering the reconfiguration of the system.

2.2.1.1 The WHEN Statement

Originally, the syntax for incorporating conditional dependence of some equations of a model in an equation-based environment was suggested by Piela (1989). That syntax is very similar to the CASE equation of *gPROMS* and suffers from the same limitations. Instead, in this work we represent that conditional dependence by using an object-oriented formalism. In an object-oriented approach any real or abstract entity is considered an object and any object can be referenced by a unique identifier (Marquardt, 1996). So, for instance, in an object-oriented language representing a model superstructure, a simple equation is considered as an object and so is any submodel within the superstructure. Based on this object-oriented approach, we defined the WHEN statement in order to represent alternative configurations of a model. The syntax of such a conditional statement is:

```
definition_of_equation_1;
definition_of_model_1;
      ⋮
definition_of_model_n;

WHEN (list_of_variables)
      CASE list_of_values_1:
                        USE identifier_of_equation_1;
      CASE list_of_values_2:
                        USE identifier_of_model_1;
      ⋮
      OTHERWISE:
                        USE identifier_of_model_n;
END WHEN;
```

The following are observations about the previous definition:

1. The declaration of the objects referenced within the CASEs of the WHEN statement is independent (outside) of such a statement. As mentioned before, the solution algorithms of conditional models require the system to have available the data structures of all those objects.
2. A list of variables is used to define the domain of validity of each of the alternative configurations. The variables in that list can be of any type among boolean, integer or symbol or any combination of them. By doing that, we place problems like logic based modeling and MINLP formulations within our scope of application.
3. Practically speaking, to "USE" an object means that the variables and equations contained in that object become an active part of the system of nonlinear equations representing the current configuration of the problem.
4. Complex reconfigurations of the problem are readily represented because of the object oriented approach of the statement.

The syntax of the WHEN statement given above can represent the conditional dependence of alternative sets of equations and variables. However, it does not say anything about (and is independent of) how to represent the conditions that trigger the dynamic switching among configurations.

2.2.1.2 Logical Relations

The way in which existing solution algorithms for conditional models select a particular configuration varies. MINLP algorithms employ the manipulation of binary variables. Some other approaches use logic for improving the solution of conditional models (Raman and Grossmann, 1994; Turkay and Grossmann, 1996). Also, the boundary crossing algorithm given by Zaher (1991) expresses the truth value of inequality constraints as boolean values of logical conditions.

Here we describe our approach to the incorporation of logical relations as a declarative modeling tool. This incorporation is intended to provide an equation-based environment with the ability to deal with the logic based formalisms required by most of the solution algorithms mentioned above.

The syntax for the representation of logical relations is rather simple:

```
logical term <==> logical term;
logical term  ==> logical term;
```

The symbols "<==>" and "==>" indicate that we have a logical relation between the two logical terms. In each of the two terms, logical operators among boolean variables such as AND, NOT, and OR are allowed.

The symbol "<==>" implies equality between the logical terms. Equality in a logical equation can also be interpreted as an *if and only if* implication between two logical terms expressed in clausal form. On the other hand, the symbol "==>" should be interpreted as the one sided implication *if* between the two terms of the logical relation. Also, it should be noted that we can express any logical clause using the proposed syntax by simply writing the clause in one of the terms and the constant boolean value TRUE in the other term of the logical equality.

In this implementation of logical relations, we maintain the equation-based philosophy. That is, the user states the logical relations that must be true at the solution to the problem but not how to solve them. Each logical equation has a residual attached to it. This residual will indicate if the expression is satisfied or not. Thus, this interpretation requires that we provide a solver that knows how to deal with logical relations. Such a solver should strive for the residuals of such equations to be true as it looks for a consistent set of values of the boolean variables in the problem.

Hence, if a consistent set of values of the boolean variables is available after each iteration, then, by checking the value of the appropriate boolean variables, an automatic change of the structure of a conditional problem is possible in an iterative solution scheme.

2.2.1.3 The CONDITION Statement

We have already stated that, in some solution algorithms for conditional models, the truth value of a boolean variable may depend on constraints expressed in terms of the real variables of the model. For this reason we propose the definition of the CONDITION statement and the SATISFIED logical operator as a complement to the implementation of logical relations. The syntax of this modeling tools is as follows:

```
CONDITION
   identifier_1: real_expression;
END CONDITION;

boolean_variable <==> SATISFIED(identifier_1,tolerance);
```

A real expression is defined and labeled inside the CONDITION statement and then the logical operator SATISFIED gives the truth value of such an expression (the residual of the real expression is compared against the tolerance defined in the SATISFIED operator). The benefits of the CONDITION statement are:

1. It contributes to the separation of equations into those given in terms of real variables and those given in terms of boolean variables, making the declarative code easier to read and understand.
2. It provides a simple way of saying that the relations defined within the statement are not going to be solved. Those relations are not a part of our nonlinear system of equations but are only used as expressions with a truth value associated to them.
3. it avoids relations containing implicit relations. Since the logical expressions are decoupled from the expressions on which they depend, it makes the life of a compiler easier.

As a summary of section 2.2.1, the WHEN statement provides an efficient means to declare alternative modeling configurations, while the conditions triggering the reconfiguration of the system can be represented by the use of logical relations and the CONDITION statement. When working together, these modeling capabilities meet the representation needs of all of the solution algorithms for conditional models we have found reported in the literature.

2.2.2 CONDITIONAL COMPILATION

Aside from the flexibility that conditional statements such as the WHEN statement give to the configuration of a model structure, another application of conditional tools is the economy of programming. An example commonly occurring in chemical engineering is the selection of the thermodynamic model to be used for equilibrium calculations. In general, it is convenient to code all of the alternative methods so that, depending on the species appearing in the equilibrium system, we can select the most appropriate method.

In these kinds of problems, the decision as to which configuration should be used has to be made at the moment in which the model is created and not during the solution process. Accordingly, what we require is building only the appropriate configuration of the problem rather than having available all the possible configurations.

Here we propose a modeling tool to incorporate conditional compilation into an equation-based environment, the SELECT statement. While this conditional tool is flexible enough to represent all of the alternatives, its presence will indicate that only those alternatives consistent with the model data will become available after the process of instantiation of the model.

The following is the syntax proposed for the conditional compilation tool:

```
defintion_of_constants;
assignment_of_constant_values;

SELECT (list_of_constants)
      CASE list_of_values_1:
                        list1_of_declarative_statements;
      CASE list_of_values_2:
                        list2_of_declarative_statements;
       ⋮
      OTHERWISE:
                        listn_of_declarative_statements;
END SELECT;
```

Even though the syntax for the SELECT statement is similar to that described for the WHEN statement, some important differences can be identified:

- In the WHEN statement, the declaration of the object is external to the conditional statement since of all the alternatives are going to be created anyway. On the contrary, in the SELECT statement the actual declaration of an object (or any other declarative statement affecting objects) is done within each CASE of the statement, explicitly discriminating among alternative configurations.

- The selection among alternatives in the SELECT statement depends on constant booleans, integer or symbols. Since these values imply a one time structural decision, they must not be modified during the solution of the problem.

Summarizing, the SELECT statement provides the capability of conditional compilation. It allows the representation of all the structural alternatives. However, since only the desired data structure is created, it does not affect the computational requirements of the model.

2.2.3 CONDITIONAL EXECUTION OF PROCEDURAL CODE

Because of the use of conditional statements in the declarative description of a model, a similar feature must also exist to give the user the ability to program the conditional execution of methods. For instance, each alternative configuration of a model may require different initialization and a different selection of the independent variables for the solution process. Hence, we propose a procedural conditional statement as follows:

```
SWITCH (list_of_variables)
      CASE list_of_values_1:
                        list1_of_procedural_statements;
      CASE list_of_values_2:
                        list2_of_procedural_statements;
       ⋮
      OTHERWISE:
                        listn_of_procedural_statements;
END SWITCH;
```

Basically, this SWITCH statement has the same application as the conditional statements that already exist in procedural modeling languages such as C and FORTRAN. Procedural statements do not involve new object definitions, they are only useful for the numerical processing of objects already created.

2.3 DETAILS OF A COMPUTATIONAL IMPLEMENTATION

The language tools introduced in the previous section provide a general framework for the representation of conditional models. In this section, we present an overview of the details that we had to address in order to create a prototype for testing the scope and application of the modeling tools.

Some of the main issues in the computer implementation of the conditional tools are:

- The implementation of the WHEN statement must provide an efficient means to generate all of the possible alternative configurations of the problem. That is, the combinatorial nature of the problem must be encapsulated without being memory intensive.
- The solution to a conditional model will reduce to solving a system of equations in which the variables and equations of the system may constantly change during the solution process. The issue here is how we are going to supply a conditional modeling solver with the correct set of variables and equations.
- The development of an approach to the implementation of conditional compilation in a declarative modeling language is a hard problem by itself.

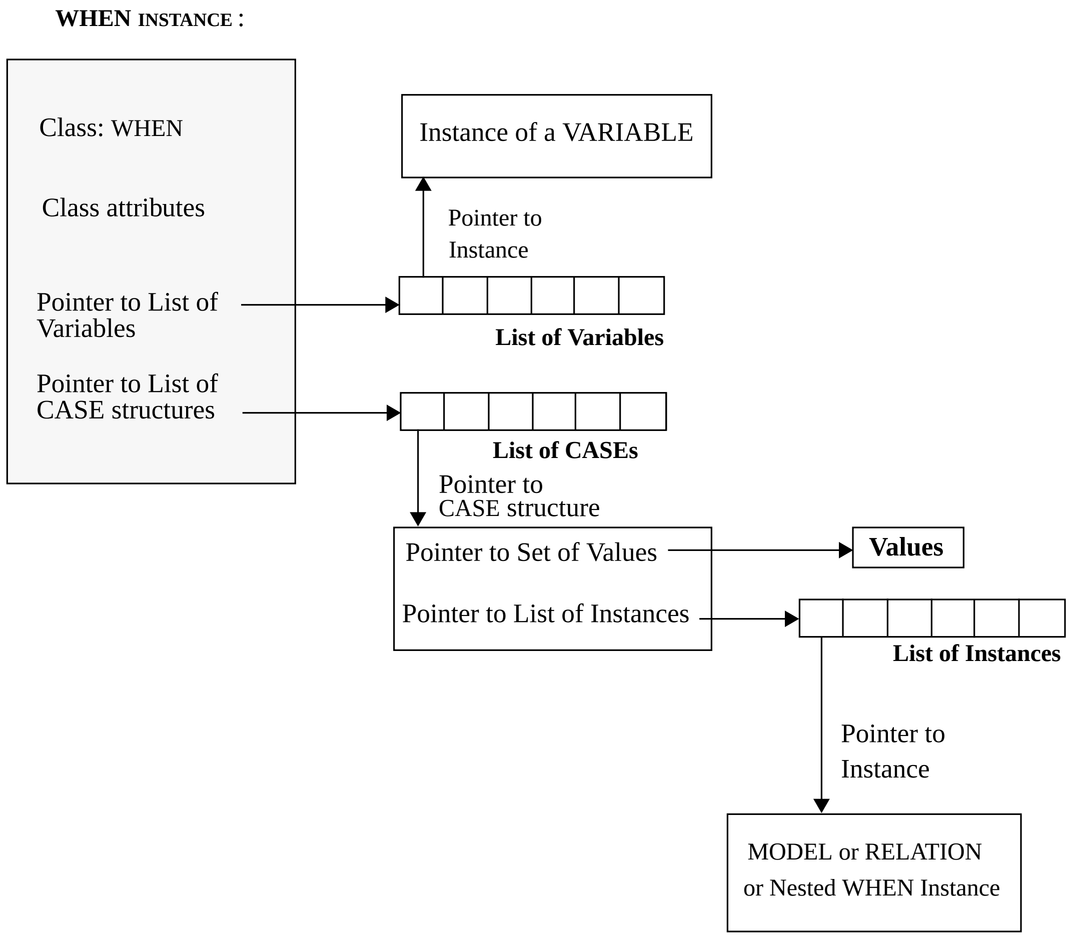
2.3.1 IMPLEMENTATION OF THE WHEN STATEMENT: THE WHEN CLASS

We follow the object oriented philosophy in order to implement a computer tool for the conditional configuration of a model structure. In a typical object-oriented modeling environment, all objects which share the same set of attributes can be viewed as an instance of a class (or type). Hence, each model is a structured class built hierarchically from instances of other models or elementary classes.

An early approach for the implementation of conditional modeling tools was described by Epperly (1988). He proposed to build a complete instance tree for each CASE within the conditional statement. However, he also recognized the combinatorial nature of that approach that makes it unacceptable; for example, for a type containing two conditional statements each having three CASEs, nine complete instance trees would be created.

In this work, we introduce the definition of an elementary class, the WHEN class. Instances of this class allow us to create a single instance tree in which all the structural alternatives are embedded. Figure 2-1 shows our approach to the implementation of a WHEN instance. Essentially, a WHEN instance is constituted by two lists of pointers: a list of pointers to instances of the conditional variables on which the WHEN statement depends and a list of pointers to CASE structures. Each CASE structure contains a list of values and another list of pointers to instances. The instances in the list of instances associated to a CASE correspond to the objects (relations, arrays of relations, models, array of models) that will be "active" when the values of the CASE list of values matches the current values of the conditional variables.

Figure 2-1 WHEN instance implementation.



With this implementation, the data structures required for all the alternative configurations are available (*i. e.*, all the objects referenced in each CASE are compiled). By visiting the instance tree and analyzing all the WHEN statements in it, we can set as "active" only the parts of the problem corresponding to the configuration consistent with the current values of the conditional variables.

2.3.1.1 Feeding a Conditional Model to a Nonlinear Solver

The implementation of the WHEN statement in an equation-based environment allow us to have available the data structure for all the variables and equations of the conditional model. Therefore, the first step in the implementation of a conditional solver is to decide how the solver is going to be fed with the correct structure and how to change this structure efficiently in an iterative solution scheme. To perform this task, we use the notion of active equation and active variable. By active we mean "it is part of the problem currently being solved." Computationally speaking, to set a relation as active or inactive implies a simple bit operation. The following steps provide a mechanism to select the structure of the problem:

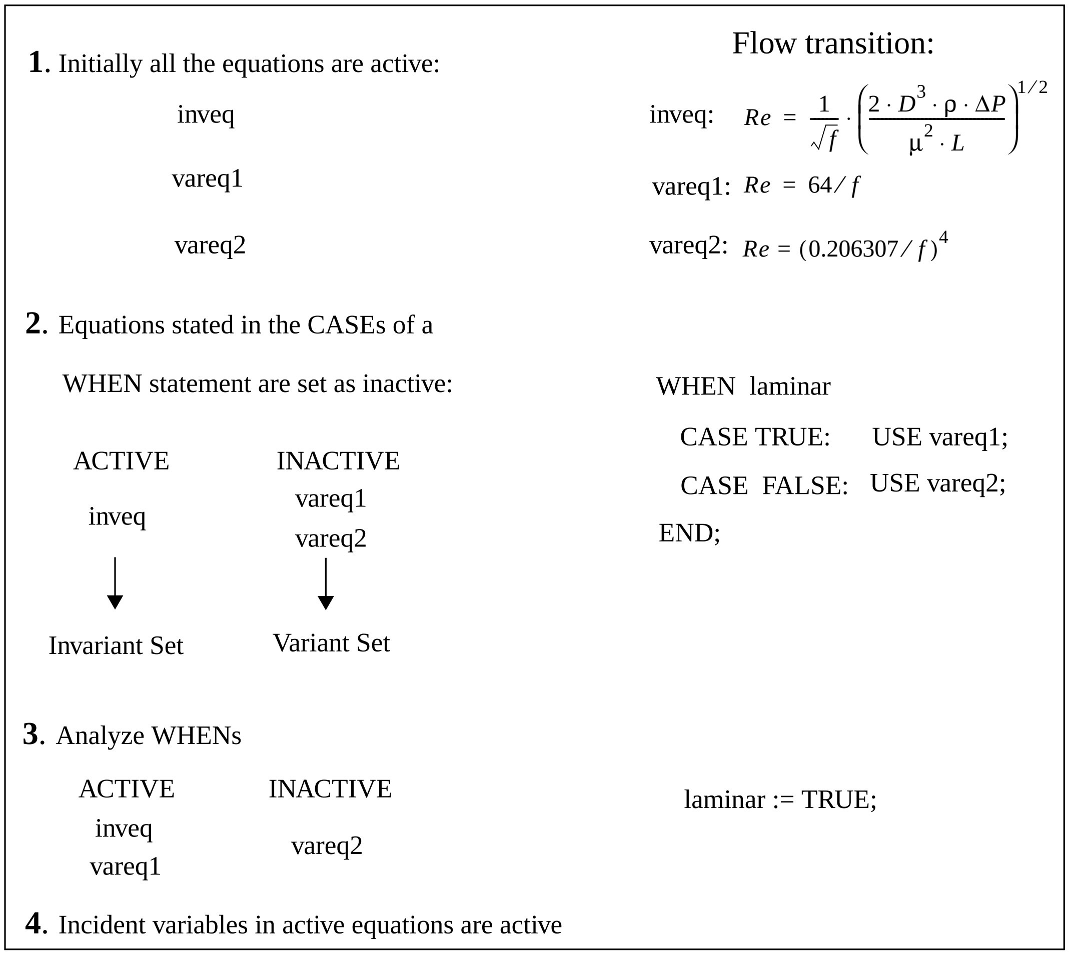
1. Initially, we consider all the equations resulting from the compilation as active.
2. Then, we set as inactive all of the equations referenced within any WHEN statement. The equations set as inactive in this step constitute the variant set of equations. All the equations which remain active constitute the invariant set of equations.
3. We analyze the WHEN statements. According to the current values of the variables on which each WHEN statement depends, we determine which of its CASEs applies. The equations stated within such a CASE are set as active.
4. All the variables incident in the active set of equations are active. The current problem to be solved consists of the active set of equations and variables.

Figure 2-2 shows the application of the previous steps to the example of the fluid flow transition. The mechanism outline above is independent of a particular solver or solution algorithm. However, we must emphasize that a change in the configuration of the problem during an iterative solution process may also cause a change in the partitioning of the variables and equations. Therefore, any solver using a partitioning mechanism must account for such a possibility.

2.3.2 CONDITIONAL COMPILATION: THE SELECT STATEMENT

During the process of instantiation of a model, it is necessary to differentiate between those objects that are not instantiated because they are defined within nonmatching CASEs of a SELECT statement from those objects that are not instantiated because of a deficiency in their declarative definition. In order to do that, we define an elementary "dummy" class.

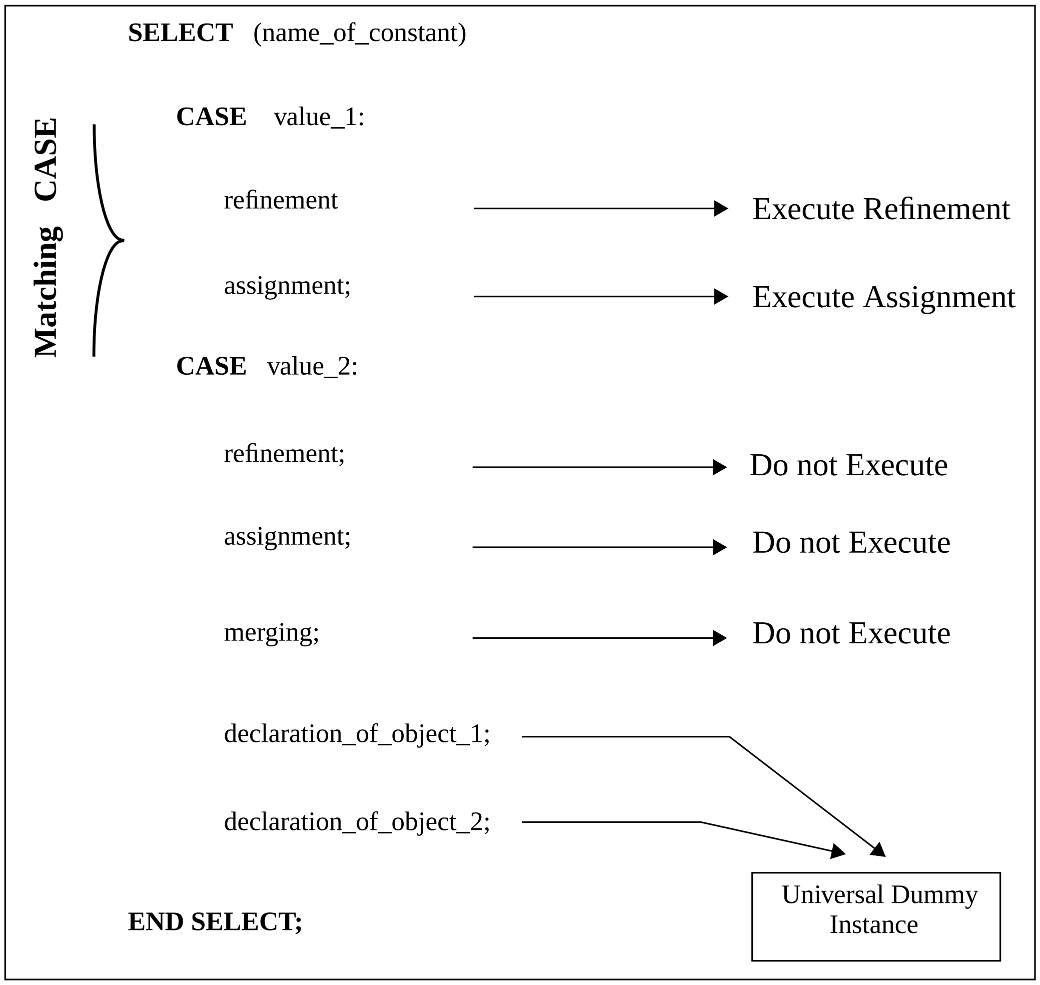
Figure 2-2 Preparing a conditional model to send to a nonlinear solver.


1. Initially all the equations are active:
inveq
vareq1
vareq2
Flow transition:
inveq: $Re = \frac{1}{\sqrt{f}} \cdot \left(\frac{2 \cdot D^3 \cdot \rho \cdot \Delta P}{\mu^2 \cdot L}\right)^{1/2}$
vareq1: $Re = 64/f$
vareq2: $Re = (0.206307/f)^4$
2. Equations stated in the CASEs of a WHEN statement are set as inactive:
ACTIVE
inveq
INACTIVE
vareq1
vareq2
Invariant Set
Variant Set
WHEN laminar
CASE TRUE: USE vareq1;
CASE FALSE: USE vareq2;
END;
3. Analyze WHENs
ACTIVE
inveq
vareq1
INACTIVE
vareq2
laminar := TRUE;
4. Incident variables in active equations are active

We find that it is necessary to build only one instance of this class, such that the dummy instance becomes a place holder for all the objects defined in all the nonmatching CASEs of the SELECT statements.

Figure 2-3 shows a graphic explanation of the process of instantiation in conditional compilation. Statements in the nonmatching CASEs of a SELECT statement that do not involve the creation of a new object are interpreted to determine if they are syntactically correct, but the resulting compiler actions are simply NOT executed (*i.e.* assignments, refinements, merging, etc.). On the other hand, statements in the matching CASEs lead to full compilation as if they were defined outside the SELECT statement.

Figure 2-3 Instantiation process in conditional compilation.



2.4 EXAMPLES OF APPLICATION

One of the goals of our research group has been to improve one's ability to develop and solve process models. The main result of this effort has been the development of ASCEND (Piela *et al*., 1991). ASCEND is both an object-oriented mathematical modeling environment and a solving and debugging engine. Westerberg *et al.* (1994) discuss the essential features of the current ASCEND system and present several ways in which we can improve it to solve larger models and to increase its scope as a modeling environment. We implemented the conditional modeling tools explained above into the ASCEND modeling environment. In this section, we show some examples to illustrate the expressiveness of the new modeling capabilities.

2.4.1 THE WHEN STATEMENT

Figure 2-4 shows two partial ASCEND models in which the WHEN statement is used, each in a different manner.

Figure 2-4 Two different applications of the WHEN statement.

```
      (a)
method            IS_A symbol;
simplified_flash  IS_A VLE_flash;
rigorous_flash    IS_A td_VLE_flash;

WHEN (method)
      CASE 'rigorous':
            USE rigorous_flash;
      CASE 'simplified':
            USE simplified_flash;
END WHEN;
```

```
            (b)
laminar     IS_A boolean;
Re,f        IS_A factor;

eq1: Re = 64/f;
eq2: Re = (0.206307/f)^4;

WHEN (laminar)
      CASE TRUE:
            USE eq1;
      CASE FALSE:
            USE eq2;
END WHEN;
```

In case (a) the value of the symbol ‘method’ is used to select between two alternative configurations of the problem, a flash calculation assuming constant relative volatility (VLE_flash) or a flash calculation using a more rigorous thermodynamic calculation (td_VLE_flash). Which option is going to be used and, therefore, the value of the symbol ‘method’, is a user decision. For example, the user may select the simplified model when looking for good initial values for the variables and then switch to the rigorous model simply by changing the value of the symbol ‘method.’ Once a configuration has been selected, it will be kept unless the user decides to change it. Note that the user does not have to recompile the model to switch. Since the system has compiled the data structure for both of the problems, the user can readily switch back and forth between the simplified model and the more complicated one.

On the contrary, in case (b) we cannot expect the boolean value of the variable laminar to be a user decision. Its value will depend on the value of the Reynolds number which is an unknown whose value is being computed in the problem. The value of the variable laminar will be the truth value of the expression $Re \leq 2100$. In an iterative solution scheme, we expect the value of the boolean variable 'laminar' to change, and, therefore, so will the structure of the system of equations that we have to solve.

Another example of the application of the WHEN statement is the synthesis of process networks using superstructure optimization. We developed a simplified model for the superstructure given in Figure 2-5, taken from the work of Turkay and Grossmann (1996). In this example, there are two alternative feedstocks, two possible choices of the reactor and two choices of the compression systems. Hence, there are 4 structural decisions, and, therefore, there are $2^4 = 16$ feasible configurations for the problem.

All 16 configurations are encapsulated in one ASCEND model containing 4 WHEN statements which depend on the value of 4 boolean variables. Figure 2-6 shows this model. The procedural section of the model and the model for each unit operation have been omitted for simplicity. Note that, for each structural decision, a WHEN statement allows the selection of either of two alternatives (a cheap reactor or a expensive reactor, for instance); however, the WHEN statements do not allow the selection of both alternatives simultaneously. Such a case would require the definition of splitters (for the input streams of the conditional units) and mixers (for the output streams of the conditional units) which, in the context of an equation-based approach, are generally considered as separate unit operations. Also, note the way in which the linking is done for the conditional units in the flowsheet. Consider, for instance, the case of the selection between an expensive reactor (r2) and a cheap reactor (r1) illustrated in Figure 2-7(a). The input streams of the two reactors are merged with the output stream of heater 1 (h1). That is, h1.output, r1.input and r2.input are merged (the ARE_THE_SAME operator in the ASCEND modeling language represents a merging operation). In practice, such a merging operation ensures that, regardless of which reactor we select (cheap or expensive), an input stream to the reactor always exists, and it is the same as the output stream from the heater 1. Similarly, a merging operation is also defined among the output streams of the reactors and the input stream to the expansion valve (v1). The combination of this merging of input and output streams with the procedure described in section 2.3.1.1 allows us to select a particular configuration of the flowsheet, as illustrated in Figure 2-7(b).

We have tested the mechanism to pass the correct submodel to a solver suggested in section 2.3.1.1 on several problems. As an example, we applied it to the system presented in Figure 2-5 and Figure 2-6. The value of the four boolean variables determine the structure of the problem. The values of the boolean variables can be defined interactively by the user, but they also can be defined by some logic inference algorithm which would allow the automatic change of the structure of the problem. In our model of the superstructure shown in Figure 2-5, the nonlinear system contains 137 invariant equations and 68 variant equations for a total of 205. The configuration defined by one of the feeds,

Figure 2-5 Superstructure taken from Turkay and Grossmann (1996).

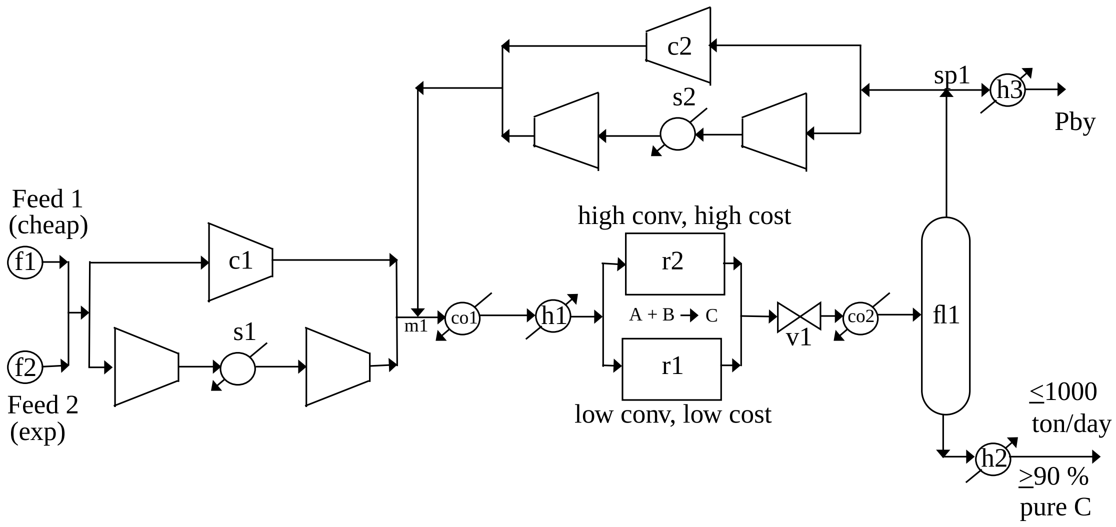
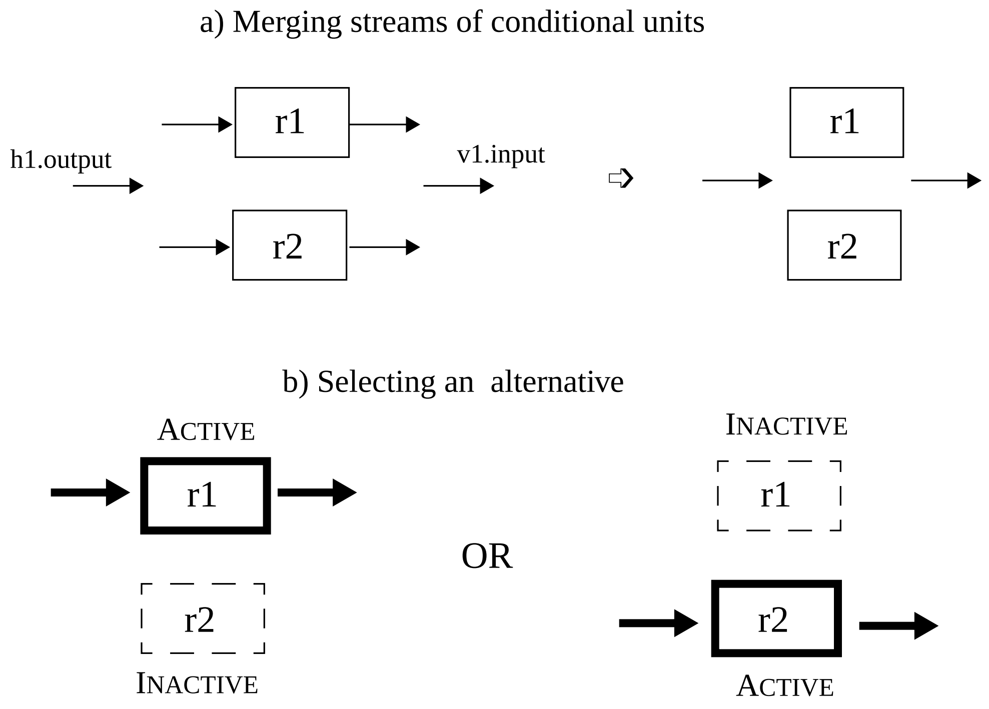


Figure 2-6 ASCEND model for the superstructure of Figure 2-5.

```
MODEL flowsheet;
(* units *)
       f1                             IS_A cheap_feed;
       f2                             IS_A expensive_feed;
       c1,c2                          IS_A single_compressor;
       s1,s2                          IS_A staged_compressor;
       r1                             IS_A cheap_reactor;
       r2                             IS_A expensive_reactor;
       co1,co2                        IS_A cooler;
       h1,h2,h3                       IS_A heater;
       fl1                            IS_A flash;
       sp1                            IS_A splitter;
       m1                             IS_A mixer;
       v1                             IS_A expansion_valve;
(* boolean variables  *)
       select_feed1,select_single1   IS_A boolean_var;
       select_cheapr1,select_single2 IS_A boolean_var;
(* define sets  *)
       m1.n_inputs :==2;
       sp1.n_outputs :== 2;
(* wire up flowsheet *)
       f1.stream, f2.stream, c1.input, s1.input     ARE_THE_SAME;
       c1.output, s1.output, m1.feed[2]             ARE_THE_SAME;
       m1.out,co1.input                             ARE_THE_SAME;
       co1.output, h1.input                         ARE_THE_SAME;
       h1.output, r1.input, r2.input                ARE_THE_SAME;
       r1.output, r2.output,v1.input                ARE_THE_SAME;
       v1.output,co2.input                          ARE_THE_SAME;
       co2.output, fl1.feed                         ARE_THE_SAME;
       fl1.liq, h2.input                            ARE_THE_SAME;
       fl1.vap, sp1.feed                            ARE_THE_SAME;
       sp1.out[1], h3.input                         ARE_THE_SAME;
       sp1.out[2],c2.input, s2.input                ARE_THE_SAME;
       c2.output, s2.output,m1.feed[1]              ARE_THE_SAME;
(* Conditional statements *)
       WHEN (select_feed1)
         CASE TRUE:
              USE f1;
         CASE FALSE:
              USE f2;
       END WHEN;
       WHEN (select_single1)
         CASE TRUE:
              USE c1;
         CASE FALSE:
              USE s1;
       END WHEN;
       WHEN (select_cheapr1)
         CASE TRUE:
              USE r1;
         CASE FALSE:
              USE r2;
       END WHEN;
       WHEN (select_single2)
         CASE TRUE:
              USE c2;
         CASE FALSE:
              USE s2;
       END WHEN;
END flowsheet;
```

two single-stage compressors and one of the reactors contains 169 equations, the 137 invariant and 32 of the 68 variant equations. Switching from one structure to another is done without the need to recompile and, since reconfiguring the system requires only rebuilding several list of pointers, it is computationally very efficient.

Figure 2-7 Linking of conditional units


a) Merging streams of conditional units
h1.output
r1
r2
v1.input
r1
r2
b) Selecting an alternative
ACTIVE
r1
r2
INACTIVE
OR
INACTIVE
r1
r2
ACTIVE

2.4.2 LOGICAL RELATIONS

Figure 2-8 illustrates two examples of the syntax we have used for the implementation of logical relations. Example (a) is simply to show the use of the logical operators AND, OR and NOT. Example (b) is a more complete version of the model presented in Figure 2-4(b). In this case we note that the value of the boolean variable ('laminar') depends on the truth value of a real expression (cond). In this way, since the WHEN statement depends on the boolean variable ‘laminar’, a change in the truth value of the real expression triggers a change in the configuration of the system.

Figure 2-8 Examples of the syntax for the implementation of logical relations.

```
      (a)                                          (b)
                                    laminar     IS_A boolean;
                                    Re,f,k      IS_A factor;
valve_open        IS_A boolean;
pump_on           IS_A boolean;     CONDITION
full_tank         IS_A boolean;           cond: Re <= 2100;
                                    END;
valve_open<==>pump_on AND
              NOT(full_tank);       laminar <==> SATISFIED(cond,1e-08);

                                    eq1: Re = 64/f;
                                    eq2: Re = (0.206307/f)^4;
                                    WHEN (laminar)
                                          CASE TRUE:
                                                USE eq1;
                                          CASE FALSE:
                                                USE eq2;
                                    END WHEN;
```

2.4.3 THE SELECT STATEMENT

Figure 2-9 shows an ASCEND model which is similar to that shown previously in Figure 2-4(a). The difference is that we use the SELECT statement rather than the WHEN statement. This time, the symbol ‘method’ is a constant, and, once it is defined, its value will not change. That value will always be a user decision. In the example of Figure 2-9, only the list of statements in the first CASE are compiled.

2.4.4 THE SWITCH STATEMENT

The use of the SWITCH statement for the conditional execution of procedural code is illustrated in Figure 2-10. In that example, if the simplified flash model is used, the average value of the constant relative volatility is set to 1.5. On the contrary, if the rigorous model is used, then a procedure defining the initial values and degrees of freedom for an adiabatic operation is executed.

Figure 2-9 Application of the SELECT statement.

```
method            IS_A symbol_constant;
method :== ‘rigorous’;

SELECT (method)
      CASE ‘rigorous’:
            rigorous_flash    IS_A td_VLE_flash;
      CASE ‘simplified’:
            simplified_flash  IS_A VLE_flash;
END SELECT;
```

Figure 2-10 Application of the SWITCH statement.

```
METHODS
      METHOD values;
            RUN reset;
            SWITCH (method)
                  CASE ‘rigorous’:
                        RUN adiabatic;
                  CASE ‘simplified’:
                        ave_alpha := 1.5;
            END SWITCH;
      END values;
```

2.5 SUMMARY

We developed efficient modeling tools for the representation of conditional models in an equation-based environment and discussed the details of their practical implementation.

The incorporation of these tools into the ASCEND environment shows how the expressiveness of an equation-based modeling language increases with this extension. Even though the new modeling capabilities are independent of any numerical technique used for the solution of a conditional model, they were developed in such a way that the needs of all of the existing solution algorithms we have found in the literature will be met.

2.6 REFERENCES

Allan, B. A.; A More Reusable Modeling System; Ph.D. thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania,1997.

Barton, P. I.; The Modeling and Simulation of Combined Discrete/Continuous Processes. Ph.D. thesis, Department of Chemical Engineering, Imperial College of Science, Technology and Medicine, 1992.

Barton, P. I. and Pantelides, C. C.; Modeling of Combined Discrete/Continuous Processes. *AIChE Journal*, 40(6):966–979, June 1994.

Epperly, T. G.; Implementation of an Ascend Interpreter. Technical Report. Engineering Design Research Center. Carnegie Mellon University. Pittsburgh, PA, 1988.

Feehery, W. F. and Barton, P. I.; A Differentiation-Based Approach to Dynamic Simulation and Optimization with High Index Differential-Algebraic Equations. *Computational Differentiation*, M. Berz, C. Bischof, G.Corliss, and A. Griewank Editors, SIAM, 1996.

Grossmann, I. E. and Turkay, M.; Solution of Algebraic Systems of Disjunctive Equations. *Comput. Chem. Eng.*, 20:S339–44, 1996. Suppl. Part A.

Marquardt, W.; Trends in Computer-Aided Process Modeling. *Comput. Chem. Eng.*, 20(6):591-609, 1996.

Pantelides, C. C.; SPEEDUP-Recent Advances in Process Simulation. *Comput. Chem. Eng.*, 12(7):745–755, 1988.

Pantelides, C. C. and Barton, P. I.; Equation-Oriented Dynamic Simulation: Current Status and Future Perspectives. *Comput. Chem. Eng.*, 17S:263–285, 1993.

Piela, P., Epperly, T., Westerberg, K., Westerberg, A. W.; An Object-Oriented Computer Environment for Modeling and Analysis: The modeling language. *Comput. Chem. Eng.*, 15(1):53–72, 1991.

Piela, P.; ASCEND: An Object-Oriented Computer Environment for Modeling and Analysis. Ph.D. thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, April 1989.

Raman, R. and Grossmann, I. E.; Modeling and Computational Techniques for Logic Based Integer Programming. *Comput. Chem. Eng.*, 18(7):563–578, 1994.

Turkay, M. and Grossmann, I. E.; Logic-Based MINLP Algorithms for the Optimal Synthesis of Process Networks. Comput. Chem. Eng., 20(8):959–978, 1996.

Westerberg, A.W., Abbott, K. A., and Allan, B. A.; Plans for ASCEND IV: Our Next Generation Equational-Based Modeling Environment. Boston, Massachusetts, November 1994. AspenWorld'94.

Zaher, J. J.; Conditional Modeling. Ph.D. thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1995.

Zaher, J. J.; Conditional Modeling in an Equation-Based Modeling Environment. The

Annual AIChE National Meeting, 1991. paper 138c.

Zaher, J. J.; Conditional Programming. The Annual AIChE National Meeting, March 1993.

CHAPTER 3 STRUCTURAL ANALYSIS OF CONDITIONAL MODELS

Structural analysis is applied to exploit sparsity in the solving of a system of equations (Duff *et al*. 1989). Zaher (1995) studied the issues involved in the structural analysis of conditional models and presented a methodology to ensure consistency in a conditional model, the complexity of such an analysis being combinatorial. In that work, Zaher considered only cases in which the number of variables and equations of all the alternatives in a conditional model are the same. In this chapter, an extension to Zaher’s consistency analysis is presented. This extension allows the consistency analysis to be applied to conditional models in which the number of variables and equations for each of the alternatives may not be the same. Also, we show how, by taking advantage of the structure of the problem, it is sometimes possible to reduce the effort required by such an analysis. In particular, the cases of the existence of repeated structures and common incidence pattern among alternatives are discussed.

3.1 INTRODUCTION/MOTIVATION

Before attempting to solve a model, a structural analysis has to be performed to determine if the model formulation is well posed. Techniques are available which can be used to detect if any structural inconsistency exists among the equations of the model (Duff *et al.*,1989; Zaher, 1995). However, consistency is generally assumed and structural analysis is rarely discussed in the literature.

Still, in practice it is very difficult to create large models of equations without introducing structural inconsistencies. We believe that structural analysis is an indispensable tool in an equation-based modeling environment, and, as we show in section 3.3, conditional models make the need for this tool an even stronger requirement.

3.2 TERMINOLOGY IN STRUCTURAL ANALYSIS

We provide here a brief and general description of the terminology employed throughout this chapter. Also, we use the system of equations given in Example 3-1 to illustrate each of the definitions.

Variables appearing in an equation are said to be *incident* in that equation and incident in the problem containing that equation. Assume that m is the number of equations in a model and n is the number of variables incident in those equations. For most engineering models, $n > m$. In order to solve a problem, the problem has to be *square*; that is, the number of equations and the number of variables to be calculated in the problem has to be the same. Accordingly, in order to solve a problem containing n variables and m equations, it is necessary to provide the values of n-m variables, so that we can calculate the rest. Thus, the difference between the number of variables and the number of equations gives us the number of *degrees of freedom*, *DOF=n-m,* of the problem. The m variables to be calculated in the problem are called *dependent variables*, while the n-m variables whose values are provided by the modeler are called *independent or decision variables*. Because of structural considerations (as we explain below), not every variable can be designated an independent variable. The set of variables whose values can be provided by the modeler (that is, the set of candidates to become an independent variable) is called the *eligible set*.

EXAMPLE 3-1 A System of Equations to Illustrate the Terminology in Structural Analysis.

$$\begin{aligned} x_1 &= 1 \\ x_2 + x_4 &= 5 \\ x_3 - x_4 + x_2 &= 3 \end{aligned} \quad \Rightarrow \quad \begin{aligned} f_1 &= x_1 - 1 = 0 \\ f_2 &= x_2 + x_4 - 5 = 0 \\ f_3 &= x_3 - x_4 + x_2 - 3 = 0 \end{aligned}$$

For the system of equations in Example 3-1:

Set of incident variables in first equation = $\{x_1\}$

Set of incident variables in the problem = $I = \{x_1, x_2, x_3, x_4\}$

Number of variables in the problem =n = 4

Number of equations in the problem =m = 3

Number of degrees of freedom = DOF = 4 - 3 = 1

The equations of a model are expected to have different sets of variables incident in them. Furthermore, they are expected to involve only a few of the variables in the problem. This observation supports the idea that models are sparse. An effective representation of a sparsity pattern of a system of equations is given by an *incidence matrix*. The rows of an incidence matrix correspond to the equations of the problem. Similarly, the columns of an incidence matrix correspond the variables incident in the equations. An element in row *i* and column *j* of an incidence matrix is nonzero if and only if the variable of column *j* is incident in the equation of row *i*. The incidence matrix of the system of equations given by Example 3-1 is shown in Figure 3-1.

Figure 3-1 Incidence matrix of Example 3-1.

$$\begin{array}{c|cccc} & x_1 & x_2 & x_3 & x_4 \\ \hline f_1 & \blacksquare & & & \\ f_2 & & \blacksquare & & \blacksquare \\ f_3 & & \blacksquare & \blacksquare & \blacksquare \end{array}$$

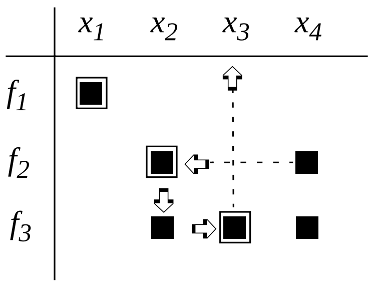
Structural inconsistencies are detected by using an incidence matrix to perform an *output assignment*. An output assignment is the process of assigning each equation to one of its incident variables. The structural *rank* is the largest number of equations which can be assigned such that no two equations are assigned to the same variable. If *rank=m*, the system of equations is *structurally consistent*. If, on the other hand, *rank<m*, the equations are guaranteed to be singular. An output assignment for the system given in Example 3-1 is shown in Figure 3-2.

Figure 3-2 An output assignment for the system defined in Example 3-1.

$$\begin{array}{c|cccc} & x_1 & x_2 & x_3 & x_4 \\ \hline f_1 & \boxed{\blacksquare} & & & \\ f_2 & & \boxed{\blacksquare} & & \blacksquare \\ f_3 & & \blacksquare & \boxed{\blacksquare} & \blacksquare \end{array}$$

Besides the detection of structural inconsistencies, the output assignment also provides a solid basis for finding a consistent partitioning of the variables. After a structurally consistent output assignment is achieved, the variables which are assigned make up a consistent set of dependent variables, while the variables left to be assigned make up a set of independent or decision variables. However, the partitioning of the variables is not unique and a generalized criterion to select the best choice does not exist. Hence, it is important to generate and make available all of the choices to the modeler. All the candidates to become independent variables can be found from an initial output assignment by following *Steward paths* (Westerberg *et al.*, 1979). A Steward path starts from an unassigned independent variable and then moves horizontally to an assigned variable, then vertically to an unassigned variable, then horizontally to an assigned variable, etc., until the path terminates, always on an assigned variable. The variables encountered along each path are marked as being eligible. Thus, all the variables gathered while transversing all of the Steward paths constitute the eligible set of variables in the problem being analyzed. A Steward path is illustrated in Figure 3-3. The combined use of an output assignment and Steward paths to obtain the eligible set is called an *eligibility analysis*.

Figure 3-3 A Steward path based on the output assignment of Figure 3-2.



The eligible set of variables for the problem of Example 3-1 is:

$$\textit{Eligible set} = \textit{Set of variables eligible to be chosen as decisions} = E = \{x_2, x_3, x_4\}$$

For the system of equations given in Example 3-1, selecting one of the variables in the eligible set E as being a independent variable results in a square structurally consistent system of equations.

3.2.1 NOTATION

The following notation is used in the remainder of this chapter. For an alternative set of equations *i* where $i \in \{1 \ldots s\}$, and *s* is the number of alternatives in a conditional model:

E_i	Eligible set	Set of variables eligible to be chosen as independent variables in the alternative *i*.
I_i	Incidence set	Set of variables incidents in the equations constituting the alternative *i*.
M	Maximal set	Union of the incidence sets of all of the alternatives.
DOF_i	Number of degrees of freedom left to be assigned.	
e	Intersection of the eligible sets of all the alternatives.	

While assigning degrees of freedom in a structural analysis, every time that a variable is chosen to be an independent variable, the elements change in the eligible set for the selection of the remaining degrees of freedom. The number of elements in the new eligible set is at least one less than the previous set. Moreover, the new eligible set is always a subset of the eligible set previous to the selection of the independent variable. For that reason, we use the index *k* to indicate a *k-th* step in the selection of the independent variables while performing structural analysis. Note that the sets I_i and *M* are independent of this index *k*, while the sets E_i and their intersection *e* change at each step *k*

k	*k-th* step in the assignment of the degrees of freedom.

3.2.1.1 Set Operators

$\cup$	Union	$A \cup B$ is all elements either in *A*, *B*, or both.
$\cap$	Intersection	$A \cap B$ is all elements in both *A* and *B*.
\	Minus	$A \backslash B$ is all elements from *A* not in *B*.

3.3 STRUCTURAL CONSISTENCY

Duff *et al*. (1989) and Zaher (1995) describe algorithms for the systematic structural consistency analysis in conventional models. They give a step by step procedure to:

- Generate an incidence matrix.
- Perform an output assignment in order to test structural consistency.
- Collect all the eligible variables by following all the Stewards path in a problem.

The interested reader may refer to those works for a detailed description of the procedures. Our attention in the rest of this chapter is focused in the structural consistency of conditional models.

3.3.1 CONDITIONAL MODELS

In conditional models, the sparsity pattern is expected to change from one alternative set of equations to another. This implies that a consistent set of independent variables for one alternative set of equations may not be valid for another one.

Zaher (1993, 1995) also addressed the structural analysis of conditional models. A necessary condition for structural consistency in conditional models is that each of the alternative sets of equations must be structurally consistent. Hence, consistency of a conditional model is assessed by finding at least one consistent partitioning of the variables (independent-dependent) such that output assignment of all of the equations in each alternative can be performed. This requirement makes the problem combinatorial, since we have to perform the analysis for all of the alternative sets of equations which can be generated from a conditional model expressed disjunctively. The following is an abbreviated description of an algorithm for finding a set of independent variables consistent with all the alternative sets of a conditional model. For a detailed description, see Zaher (1995). It is assumed that there is a nonzero number of degrees of freedom in the problem:

1. Each of the alternative sets of equations is first arbitrarily output assigned.
2. For each output assignment, the set of variables eligible to become independent (E_i^k) is

generated. In general, the eligible sets generated for each of the alternatives are different.

3. Since, it is necessary (but not sufficient) that a variable which is eligible in the context of the overall problem must be eligible for each alternative, we next find the intersection of the eligible sets generated in step 2:

$$e^k = E_1^k \cap E_2^k \cap \dots E_s^k = \bigcap_j^s E_j^k \tag{3.1}$$

where s the number of alternatives in the conditional model.

4. We tentatively select a variable from the intersection set generated in step 3 to be an independent variable. After this step, the number of independent variables left is reduced by one, and the process is repeated from step 1 using the remaining dependent variables.
5. Consistency is achieved only if a sequence is found which allows an eligible variable to be selected for each degree of freedom. Therefore, we backtrack anytime we fail to complete such a sequence.

3.3.1.1 Limitations of Zaher's Consistency Analysis

Consider a conditional model in which 's' alternative sets of equations can be generated. In his work, Zaher only addressed the case in which the variables incident in each of the alternatives is the same,

$$I_1 = I_2 = \dots = I_s = I$$

Given that condition, the variables common to all of the eligible sets of each alternative set of equations (e^k) can be regarded as eligible to become independent in the context of the overall conditional model.

In a general situation, however, the number of equations in each alternative of a conditional model may change and so may the incidence set of variables.

3.4 EXTENSION OF THE CONSISTENCY ANALYSIS FOR CONDITIONAL MODELS

For the case developed by Zaher in which the incident variables of all the alternatives are the same, the result of applying equation (3.1) is the elimination of all those variables which are eligible to be chosen as independent variables in some alternatives but non eligible to be chosen as independent variables in some other alternatives. That is readily accomplished by using the intersection of the individual eligible sets since all the variables are incident in all the alternatives.

For the general case, however, since we expect

$$I_1 \neq I_2 \neq I_3 \ldots \neq I_s$$

we cannot use the intersection of the eligible set of each alternative to generate the eligible set for the overall conditional model. If we would do that, we would immediately remove variables which are not incident in some of the alternatives, since they would not be eligible for an alternative in which they are not incident.

A detailed derivation of an equivalent to (3.1) when the alternatives of a conditional model have different incident variables is presented in Appendix A. In Appendix A, we show that, in general, for any alternative $i \in \{1 \ldots s\}$, the set of "truly" eligible variables (ineligible variables are eliminated) for each alternative in the context of the overall conditional problem is given by:

$$E_i^{k\prime} = E_i^k \backslash \left\{ E_i^k \cap \left[\bigcup_{j,\, j \neq i}^{s} (I_j \backslash E_j^k) \right] \right\} \quad \textbf{(3.2)}$$

and that the union of these individual sets gives the set of eligible variables from which we can safely select the independent variables of a conditional model:

$$e^{k\prime} = \bigcup_i^s E_i^{k\prime} \quad \textbf{(3.3)}$$

Hence, by using (3.3) instead of (3.1), we can apply the structural consistency algorithm described in 3.3.1 to a general conditional model having alternatives with different incident variables.

Furthermore, in Appendix A we also show that we do not have to perform the analysis for the general case as defined in (3.3). A simpler analysis can be used instead. We demonstrate that, if we augmented the eligible set of each alternative E_i^k with the non incident variables of that alternative:

$$E_i^{k''} = E_i^k \cup (M \backslash I_i) \tag{3.4}$$

and find the intersection of the augmented sets $E_i^{k''}$,

$$e^{k''} = \bigcap_i^s E_i^{k''} = \bigcap_i^s [E_i^k \cup (M \backslash I_i)] \tag{3.5}$$

then the resulting set $e^{k''}$ is equivalent to the set $e^{k'}$ given by (3.3). Recall that *M* is the maximal set of variables,

$$M = I_1 \cup I_2 \cup \ldots I_s = \bigcup_j^s I_j \tag{3.6}$$

so that $(M \backslash I_i)$ represents the set of non incidences in alternative *i*.

Therefore, the use of (3.5) instead of (3.1) also allows us to apply the structural consistency algorithm described in 3.3.1 to a general conditional model having alternatives with different incident variables. As a final result in Appendix A, we also demonstrate that both (3.5) and (3.3) reduce to (3.1) when the alternatives of a conditional model have the same incident variables.

3.4.1 THE MAIN RESULT

The main result of the analysis presented in this section is that the algorithm described in section 3.3.1, derived for a conditional model having the same incident variables in all its alternative set of equations, can still be applied for a general case in which different alternatives of a conditional model have different incident variables. In order to accomplish that, the step 3 of the algorithm for testing structural consistency given in 3.3.1 has to be modified by using (3.5) instead of (3.1).

3.4.2 AN IMPLEMENTATION

Our extension to the consistency analysis described in 3.3.1 has been incorporated as a tool within the ASCEND modeling environment to help a user to set up structurally consistent conditional models. Essentially, this implementation uses the information provided by the conditional statements described in Chapter 2 in order to generate the alternative configurations of a conditional model. Then, for each of the alternatives, we apply the eligibility analysis techniques already available for conventional models.

3.4.3 AN ILLUSTRATIVE EXAMPLE

The formulation derived in this section is illustrated with Example 3-2. In this case, a disjunctive system of equations contains two disjunctions, which results in 4 different alternative set of equations. The 4 set of equations derives from Example 3-2 are shown in Figure 3-4.

For the purpose of illustration, assume that the variables x_2, x_4, and x_7 have already been selected as independent variables. Hence, the set of independent variables, $\wp$, prior to the analysis is:

$$\wp = \{x_2, x_4, x_7\}$$

EXAMPLE 3-2 A simple disjunctive set of equations.

$$x_{18} = 1$$
$$x_1 = 0.8 \cdot x_2$$
$$x_3 = 10 - x_4$$

$$\begin{bmatrix} x_5 = x_1 + x_3 \\ x_6 = x_5 - x_1 \\ x_8 = x_7 + x_6 \end{bmatrix} \vee \begin{bmatrix} x_7 = 0.9 \cdot x_8 \\ x_9 = x_8 + x_{10} + x_3 \\ x_{10} = 40 - x_9 \\ x_{13} = x_{21} \end{bmatrix}$$

$$\begin{bmatrix} x_{12} = 3 \cdot x_{21} \cdot x_3 \\ x_{14} = x_7 + x_8 \\ x_{16} = x_{24} + x_{11} \\ x_{15} = x_{16} + x_{23} \\ x_{23} = x_4 \end{bmatrix} \vee \begin{bmatrix} x_{17} = x_7 \\ x_{19} = x_{20} \\ x_{20} = x_{22} \end{bmatrix}$$

Figure 3-4 4 Alternatives in the Example 3-2.

Alternative 1	Alternative 2	Alternative 3	Alternative 4
$x_{18} = 1$	$x_{18} = 1$	$x_{18} = 1$	$x_{18} = 1$
$x_1 = 0.8 \cdot x_2$	$x_1 = 0.8 \cdot x_2$	$x_1 = 0.8 \cdot x_2$	$x_1 = 0.8 \cdot x_2$
$x_3 = 10 - x_4$	$x_3 = 10 - x_4$	$x_3 = 10 - x_4$	$x_3 = 10 - x_4$
$x_5 = x_1 + x_3$	$x_5 = x_1 + x_3$	$x_7 = 0.9 \cdot x_8$	$x_7 = 0.9 \cdot x_8$
$x_6 = x_5 - x_1$	$x_6 = x_5 - x_1$	$x_9 = x_8 + x_{10} + x_3$	$x_9 = x_8 + x_{10} + x_3$
$x_8 = x_7 + x_6$	$x_8 = x_7 + x_6$	$x_{10} = 40 - x_9$	$x_{10} = 40 - x_9$
		$x_{13} = x_{21}$	$x_{13} = x_{21}$
$x_{12} = 3 \cdot x_{21} \cdot x_3$	$x_{17} = x_7$	$x_{12} = 3 \cdot x_{21} \cdot x_3$	$x_{17} = x_7$
$x_{14} = x_7 + x_8$	$x_{19} = x_{20}$	$x_{14} = x_7 + x_8$	$x_{19} = x_{20}$
$x_{16} = x_{24} + x_{11}$	$x_{20} = x_{22}$	$x_{16} = x_{24} + x_{11}$	$x_{20} = x_{22}$
$x_{15} = x_{16} + x_{23}$		$x_{15} = x_{16} + x_{23}$	
$x_{23} = x_4$		$x_{23} = x_4$	

Table 3-1 shows an analysis of the degrees of freedom left to be assigned for each of the alternatives. Note that the number of equations, the number of incident variables, and the number of degrees of freedom left to be assigned are different for each alternative.

Table 3-1 Degrees of freedom left to be assigned in Example 3-2.

Alternative	Number of equations	Number of incidences	Number of DOF	DOF left to be assigned
1	11	17	6	3
2	9	13	4	1
3	12	18	6	3
4	10	15	5	2

The eligible set for each of the alternatives, obtained from the eligibility analysis we described, is shown in Table 3-2.

Table 3-2 Incidence and eligible sets in Example 3-2.

Alternative	Incidences, I_i	Eligible set, E_i
1	x_1, ... x_8, x_{11}, x_{12}, x_{14},...x_{16}, x_{18}, x_{21}, x_{23}, x_{24}	x_{11}, x_{12}, x_{15}, x_{16}, x_{21}, x_{24}
2	x_1, ... x_8, x_{17}, ..., x_{20}, x_{22}	x_{19}, x_{20}, x_{22}
3	x_1, ... x_4, x_7, ..., x_{16}, x_{18}, x_{21}, x_{23}, x_{24}	x_{11}, x_{12}, x_{13}, x_{15}, x_{16}, x_{21}, x_{24}
4	x_1, ... x_4, x_7, ..., x_{10}, x_{13}, x_{17},...x_{22}	x_{13}, x_{19}, x_{20}, x_{21}, x_{22}

If we would try to apply Zaher's structural analysis in order to obtain a consistent set of independent variables for the overall problem, we would conclude structural inconsistency, since the intersection of the individual eligible sets is empty:

$$e^k = \bigcap_j^s E_j^k = \varnothing$$

Instead, we can apply either of the equations derived in this section to obtain:

from (3.3),

$$e^{k\prime} = \bigcup_{i}^{s} E_i^{k\prime} = \{x_{11}, x_{12}, x_{13}, x_{15}, x_{16}, x_{19}, x_{20}, x_{21}, x_{22}, x_{24}\}$$

or, from (3.5),

$$M = \{x_1 \ldots x_{24}\}$$

$$e^{k\prime\prime} = \bigcap_{i}^{s} E_i^{k\prime\prime} = \{x_{11}, x_{12}, x_{13}, x_{15}, x_{16}, x_{19}, x_{20}, x_{21}, x_{22}, x_{24}\}$$

Note that the result is the same, $e^{k\prime} = e^{k\prime\prime}$. By applying the consistency algorithm in 3.3.1, we obtain the following set of consistent independent variables:

$$\{x_{11}, x_{15}, x_{19}, x_{21}\}$$

Hence, if we partition the variables in Example 3-2, so that the set of independent variables is given by,

$$\wp = \{x_2, x_4, x_7, x_{11}, x_{15}, x_{19}, x_{21}\}$$

then all 4 alternatives generated from Example 3-2 are square and structurally consistent.

3.4.4 IS THE COMBINATORIAL CONSISTENCY ANALYSIS REALLY NECESSARY ?

Given the combinatorial nature of conditional models, it may happen that, during the iterative solution of one of such models, only a few of the alternatives of it are considered before converging to the solution. For that reason, a question comes to mind. Why should we look for the structural consistency of all of the alternatives in a conditional model if it is true that many of those alternatives will never be encountered during an iterative solution technique ?

First, we should recognize that it is also true that, in general, there is no way of knowing which alternatives will be visited during an iterative solution of a conditional model.

Hence, we believe that it is a valid methodology to consider the structural analysis of only those alternatives encountered during the solution of a conditional model if such procedure reaches a satisfactory result. However, such methodology cannot guarantee that a consistent set of independent variables selected for an alternative will also be consistent with respect to any another alternative visited later during the iterative solution technique.

In general, in order to ensure the structural consistency of a conditional model, the combinatorial consistency analysis must be performed.

3.5 SIMPLIFYING THE CONSISTENCY ANALYSIS OF CONDITIONAL MODELS

The analysis required for partitioning the variables in a conditional model was outlined in section 3.3.1. We see the most serious disadvantage of this analysis to be the combinatorial nature of the search consistency algorithm, which requires the analysis of all of the alternatives every time that a selection of an independent variable is made.

However, we have observed special features of some problems which can contribute to simplifying the analysis. Even though they represent very particular cases, if found alone or in any combination, we can take advantage of them in order to reduce the computational effort needed to perform the analysis.

3.5.1 COMMON INCIDENCE PATTERN

It happens sometimes that the equations of some alternatives in a conditional model are different only because of the difference in value of some parameters, such as cost factors, mass balance coefficients, power of a correlation, etc. That is, the incidence pattern of those alternatives is the same. In such a case, the combinatorial structural analysis can be greatly simplified. Consider the conditional model defined in Example 3-3. In that example, there are 6 disjunctions in the problem. Hence, the number of alternative set of equations is:

$$\textit{Number of alternatives} = 2^6 = 64$$

EXAMPLE 3-3 Taking advantage of a common incidence pattern.

$$x_1 = x_6 + x_{12}$$

$$x_9 = x_{10} + x_{11}$$

$$\begin{bmatrix} x_6 = 1.15 \cdot x_7 \\ x_{10} = 0.1 \cdot x_7 \\ x_7 \le 8 \end{bmatrix} \vee \begin{bmatrix} x_6 = 1.2 \cdot x_7 \\ x_{10} = 0.2 \cdot x_7 \\ x_7 \ge 8 \end{bmatrix}$$

$$\begin{bmatrix} x_2 = 0.47 \cdot x_8 \\ x_7 = 0.75 \cdot x_8 \\ x_8 \le 10 \end{bmatrix} \vee \begin{bmatrix} x_2 = 0.45 \cdot x_8 \\ x_7 = 0.7 \cdot x_8 \\ x_8 \ge 10 \end{bmatrix}$$

$$\begin{bmatrix} x_8 = 1.8 \cdot x_4 \\ x_9 = 0.7 \cdot x_2 \\ x_4 \le 11 \end{bmatrix} \vee \begin{bmatrix} x_8 = 1.87 \cdot x_4 \\ x_9 = x_1 \\ x_4 \ge 11 \end{bmatrix}$$

$$\begin{bmatrix} x_3 = 1.15 \cdot x_{13} \\ x_{12} = 0.25 \cdot x_{13} \\ x_{13} \le 9 \end{bmatrix} \vee \begin{bmatrix} x_3 = 1.10 \cdot x_{13} \\ x_{12} = 0.3 \cdot x_{13} \\ x_{13} \ge 9 \end{bmatrix}$$

$$\begin{bmatrix} x_{11} = 0.35 \cdot x_{14} \\ x_{13} = 1.25 \cdot x_{14} \\ x_{14} \le 8 \end{bmatrix} \vee \begin{bmatrix} x_{11} = 0.3 \cdot x_{14} \\ x_{13} = 1.3 \cdot x_{14} \\ x_{14} \ge 8 \end{bmatrix}$$

$$\begin{bmatrix} x_{14} = 1.10 \cdot x_5 \\ x_5 \le 4 \end{bmatrix} \vee \begin{bmatrix} x_{14} = 1.02 \cdot x_5 \\ x_5 \ge 4 \end{bmatrix}$$

However, it is trivial to observe that, in all but one of the disjunctions, the incidence pattern is the same. As a consequence, for the purposes of structural analysis, only two different alternatives have to be considered.

$$\textit{Number of alternatives for structural analysis} = 2^1 = 2$$

Hence, for instance, the structural analysis for the problem of Example 3-3, could be simplified to one of the system of equations shown in Figure 3-5.

3.5.1.1 An Implementation

In parallel with the implementation of our extension to the consistency analysis of a conditional model, we have also implemented a computer tool whose goal is identifying all those conditional structures with the same incidence pattern. This tool has been

Figure 3-5 Taking advantage of a common incidence pattern.

$$x_1 = x_6 + x_{12}$$
$$x_9 = x_{10} + x_{11}$$

$$x_6 = 1.15 \cdot x_7$$
$$x_{10} = 0.1 \cdot x_7$$

$$x_2 = 0.47 \cdot x_8$$
$$x_7 = 0.75 \cdot x_8$$

$$\begin{bmatrix} x_8 = 1.8 \cdot x_4 \\ x_9 = 0.7 \cdot x_2 \end{bmatrix} \vee \begin{bmatrix} x_8 = 1.87 \cdot x_4 \\ x_9 = x_1 \end{bmatrix}$$

$$x_3 = 1.15 \cdot x_{13}$$
$$x_{12} = 0.25 \cdot x_{13}$$

$$x_{11} = 0.35 \cdot x_{14}$$
$$x_{13} = 1.25 \cdot x_{14}$$

$$x_{14} = 1.10 \cdot x_5$$

incorporated within the ASCEND environment. It uses the information provided by the conditional statements described in Chapter 2 (WHEN statement) in order to compare the incidence pattern of alternative configurations. This tool is applied as a step prior to the consistency analysis described in section 3.4. Hence, the consistency analysis considers only those alternatives whose incidence patterns are different, and, therefore, the combinatorial complexity of the analysis is reduced.

3.5.2 REPEATED STRUCTURES

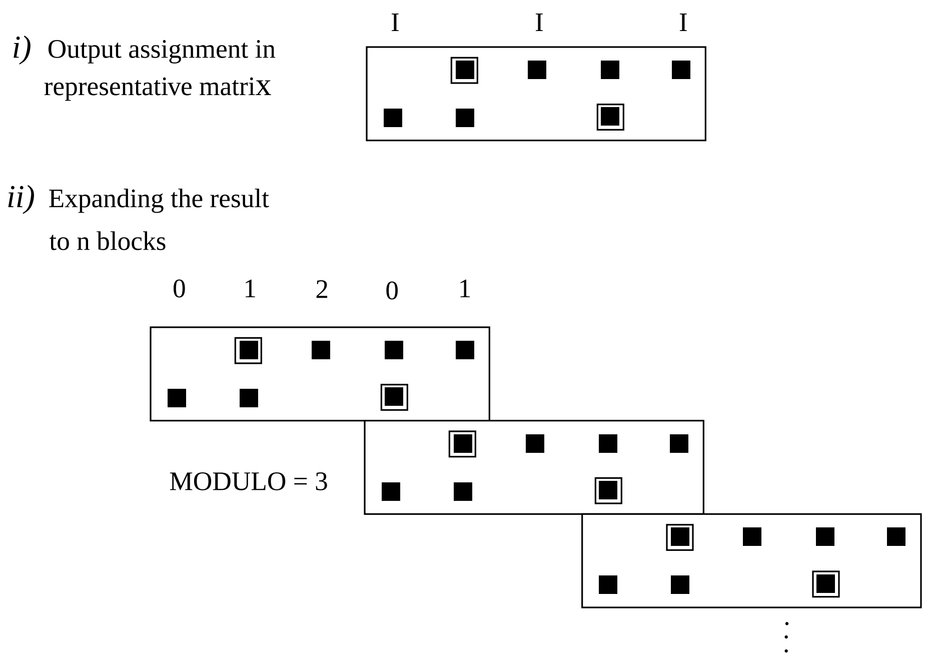
In chemical engineering design and simulation, systems containing repeated structures occur very often. Typical examples are a distillation column (n trays) and systems of equations arising in the discretization of an initial value problem. If we think of a problem in which a disjunction exists in each of the repeated structures, the combinatorial complexity of the consistency analysis would be unmanageable. However, intuition suggests that, in these kinds of problems, the degrees of freedom analysis should not be affected by the number of repeated structures and, therefore, the analysis could be simplified. Based on the work of Allen and Westerberg (1976), we show here that sometimes it is possible to take considerable advantage of the existence of repeated structures in a conditional model, reducing the effort required by the consistency analysis.

3.5.2.1 An Approach for Conventional Models

In their work, Allen and Westerberg used a representative incidence matrix to perform a systematic analysis of a conventional model containing repeated structures. They provide a criterion to decide whether or not, after a consistent output assignment has been found for the representative matrix, the result obtained for the representative matrix can be expanded to a system containing any number of the repeated structures. Consider the simple case illustrated in Figure 3-6. In this example, the output assignment of a system containing two equations has been performed, resulting in the selection of a structurally consistent set of 3 independent variables (marked as I in Figure 3-6). The problem consists in finding if we can expand this partitioning to a system containing *n* blocks of the same two equations. First, it is necessary to introduce the definition of the *modulo* of the expansion. Modulo is the number of positions that each new block added to the structure is going to move from the left-most entry (or right-most entry if the expansion is upward) of the previous block. In other words, modulo is the number of columns that each new block is going to be displaced with respect to the previous one. In the case of Figure 3-6, the modulo of the expansion is equal to 3.

Once the modulo of the expansion is known, Allen and Westerberg propose to enumerate the columns of the representative block successively from *0* to *modulo-1* until reaching the last column of the block. Then, the necessary condition for expanding the result to *n* blocks is that no two columns representing a dependent variable in the representative matrix can have the same column number. In Figure 3-6, this criterion is satisfied since the two columns representing the dependent variables has been enumerated as *1* and *0*, and, therefore, the expansion can take place. Practically speaking, what this condition means is that we cannot allow, after an expansion to *n* blocks, the overlapping of any two columns chosen as dependent variables. If we allow that, we would have a variable assigned to more than one equation, a situation that violates the structural consistency requirement explained in 3.2 (definition of the output assignment).

Figure 3-6 Expanding the result of a representative matrix.


i) Output assignment in representative matrix
I
I
I
ii) Expanding the result to n blocks
0
1
2
0
1
MODULO = 3

3.5.2.2 Equation-Based Modeling and the Modulo of Repeated Structures

In the context of an equation-based modeling, the set of overlapping variables and, therefore, the modulo of an array of repeated structures, are given by the connections among the repeated structures. In most of the existing equation-based environments currently available, there exist language constructs which allow the representation of connections defining the flow of information. So, for instance the IS operator of gPROMS (Barton, 1992) and the ARE_THE_SAME operator of ASCEND (Piela, 1989) serve this purpose.

3.5.2.3 Repeated Structures Containing Conditional Equations

As stated earlier, the combinatorial complexity of the consistency analysis of systems containing repeated structures with conditional equations would be practically unmanageable.

What we propose here is to perform the consistency analysis of this type of conditional model by using a representative structure of the problem. The difference with respect to the work presented by Allen and Westerberg is that, in our problem, the basic structure to be considered in the analysis contains conditional equations, and, therefore, a consistency analysis over all the possible configurations of the representative structure has to be performed. In other words, we combine the consistency analysis developed for conditional models with the idea of analyzing only a representative block of an array of repeated structures. Example 3-4 and Example 3-5 illustrate the application of this approach.

3.5.2.4 Illustrative Examples

Example 3-4 is used to illustrate an extreme case, in which there may be sets of nested repeated structures containing conditional equations. We use indices *n* and *m* to indicate the number of repeated structures in each of the sets.

EXAMPLE 3-4 Taking advantage of repeated structures.

$$x_1 + x_{2_{1,1}} = 4$$

$$x_{2_{1,1}} - 2 \cdot x_1 = 7$$

$$\left[\begin{array}{c} x_{2_{i,1}} - x_{3_{i,1}} + x_{4_{i,1}} = 3 \\ \left[\begin{array}{c} x_{4_{i,j}} + x_{5_{i,j}} + x_{6_{i,j}} + x_{8_{i,j}} = 9 \\ x_{6_{i,j}} + x_{7_{i,j}} - x_{3_{i,j}} - x_{5_{i,j}} = 1 \end{array}\right] \vee \left[\begin{array}{c} x_{4_{i,j}} + x_{5_{i,j}} - x_{8_{i,j}} + x_{7_{i,j}} = 8 \\ x_{6_{i,j}} + x_{4_{i,j}} - x_{3_{i,j}} = 2 \end{array}\right] \\ \forall j \in \{1 \ldots m\} \\ \left.\begin{array}{ll} x_{3_{i,j}}, x_{6_{i,j-1}} & ATS \\ x_{4_{i,j}}, x_{7_{i,j-1}} & ATS \\ x_{5_{i,j}}, x_{8_{i,j-1}} & ATS \end{array}\right\} \forall j \in \{2 \ldots m\} \end{array}\right] \vee \left[\begin{array}{c} x_{2_{i,1}} + x_{3_{i,1}} - x_{7_{i,1}} + x_{5_{i,1}} = 6 \\ \left[\begin{array}{c} x_{3_{i,j}} + x_{8_{i,j}} - x_{6_{i,j}} = 7 \\ x_{4_{i,j}} - x_{5_{i,j}} - x_{7_{i,j}} = 0 \end{array}\right] \vee \left[\begin{array}{c} x_{5_{i,j}} - x_{3_{i,j}} + x_{6_{i,j}} = 4 \\ x_{8_{i,j}} + x_{4_{i,j}} - x_{7_{i,j}} = 0 \end{array}\right] \\ \forall j \in \{1 \ldots m\} \\ \left.\begin{array}{ll} x_{3_{i,j}}, x_{6_{i,j-1}} & ATS \\ x_{4_{i,j}}, x_{7_{i,j-1}} & ATS \\ x_{5_{i,j}}, x_{8_{i,j-1}} & ATS \end{array}\right\} \forall j \in \{2 \ldots m\} \end{array}\right]$$

$$\forall i \in \{1 \ldots n\}$$

$$\left.\begin{array}{ll} x_{2_{i,1}}, x_{6_{i-1,m}} & ATS \\ x_{3_{i,1}}, x_{7_{i-1,m}} & ATS \\ x_{4_{i,1}}, x_{8_{i-1,m}} & ATS \end{array}\right\} \forall i \in \{2 \ldots n\}$$

$$x_{6_{n,m}} - x_{7_{n,m}} - x_{8_{n,m}} = 2$$

$$x_{6_{n,m}} - x_{7_{n,m}} = 4$$

For simplicity in the representation, the abbreviation ATS (ARE_THE_SAME, following the ASCEND modeling language representation) is used to express the connectivity among the variables incident in the repeated structures. The number of alternatives in problems of this nature grows very quickly with the number of repeated structures. So, for instance, if *m=5* and *n=5*, the number of alternatives is:

$$\textit{number of alternatives} = (2^5)^5 = 33554432$$

In order to structurally analyze this conditional model, we use a representative structure of the problem. For this case, this representative structure is given for a system in which *n=m=1*. The resulting system of equations is given in (3.7), where the subindexes *n=m=1* have been omitted for simplicity.

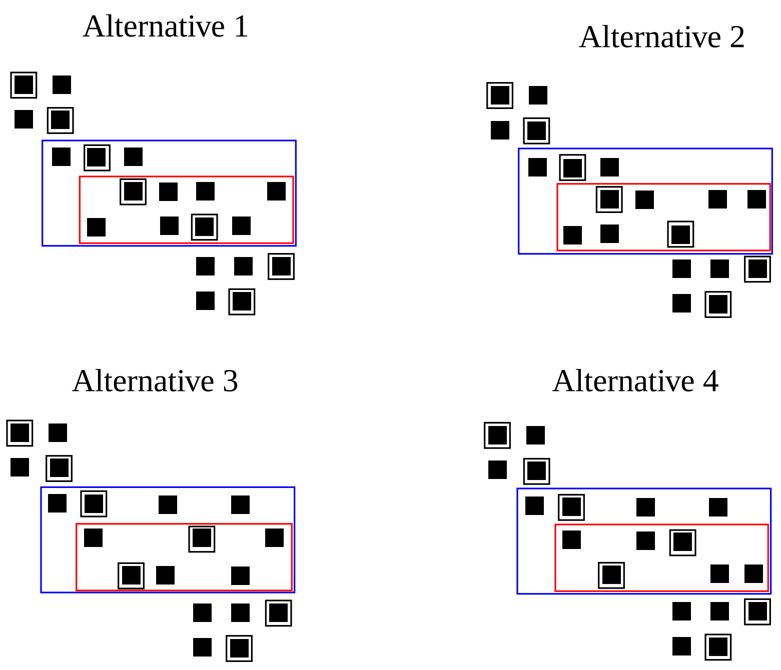
$$\begin{gathered}
x_1 + x_2 = 4 \\
x_2 - 2 \cdot x_1 = 7 \\
\begin{bmatrix} x_2 - x_3 + x_4 = 3 \\ \begin{bmatrix} x_4 + x_5 + x_6 + x_8 = 9 \\ x_6 + x_7 - x_3 - x_5 = 1 \end{bmatrix} \vee \begin{bmatrix} x_4 + x_5 - x_8 + x_7 = 8 \\ x_6 + x_4 - x_3 = 2 \end{bmatrix} \end{bmatrix} \vee \begin{bmatrix} x_2 + x_3 - x_7 + x_5 = 6 \\ \begin{bmatrix} x_3 + x_8 - x_6 = 7 \\ x_4 - x_5 - x_7 = 0 \end{bmatrix} \vee \begin{bmatrix} x_5 - x_3 + x_6 = 4 \\ x_8 + x_4 - x_7 = 0 \end{bmatrix} \end{bmatrix} \\
x_6 - x_7 - x_8 = 2 \\
x_6 - x_7 = 4
\end{gathered} \qquad \textbf{(3.7)}$$

The representative structure (3.7) contains 4 alternatives, each of them with one degree of freedom. By applying the structural consistency algorithm to the simplified problem, we obtain that the eligible set of the representative conditional model is:

$$\textit{eligible set} = \{x_3, x_4, x_5, x_6, x_7, x_8\}$$

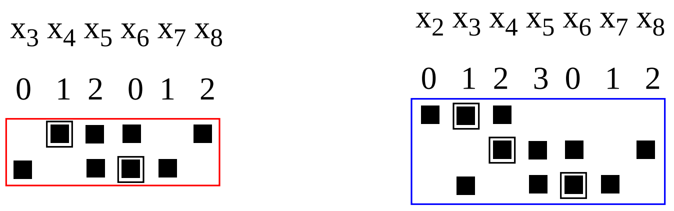
The task is to find a set of independent variables (only one variable in this example) which, for each alternative, allows us an output assignment satisfying Allen's necessary conditions for expanding the result of a representative incidence matrix. For the representative conditional structure of Example 3-4, assigning x_5 as independent variable allows the output assignments presented in Figure 3-7 for each of the alternatives.

Figure 3-7 Alternative configurations of the representative structure.



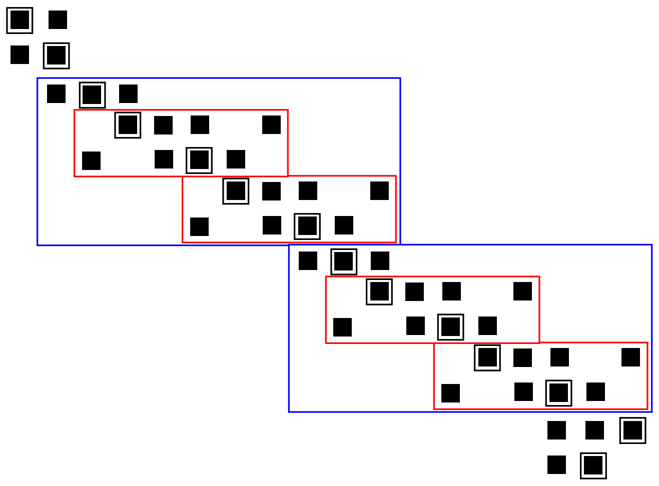
All 4 output assignments of Figure 3-7 satisfy Allen’s necessary conditions. Figure 3-8 illustrates the case of the nested blocks of alternative 1. The modulo of each block was determined by the connectivity among the repeated structures described in the formulation of Example 3-4.

Figure 3-8 Allen’s necessary conditions for alternative 1 of the representative matrix.


x_3 x_4 x_5 x_6 x_7 x_8
0 1 2 0 1 2
x_2 x_3 x_4 x_5 x_6 x_7 x_8
0 1 2 3 0 1 2

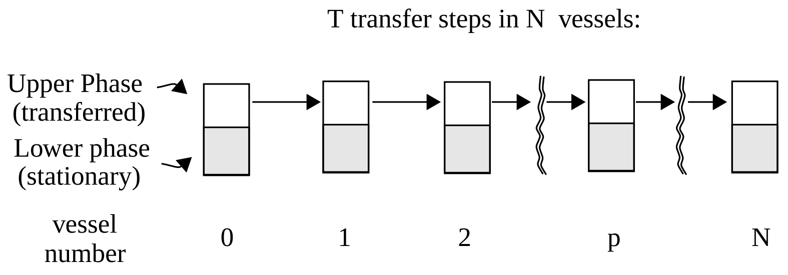
We could show that, if we expand the result of the representative incidence matrix to any number of repeated structures and in any combination, the resulting systems of equations are still structurally consistent. Figure 3-9 shows an example of this expansion. Note that each variable $x_{5_{i,j}}$ is selected as independent variable (that is, it is not assigned).

Figure 3-9 Expanding the result of a representative matrix. Expansion of alternative 1 with *n=m=2*.



The final example of this chapter, Example 3-5, corresponds to a chromatographic separation performed in CCD (Craig Countercurrent Distribution) discussed by King (1980). This example is illustrated in Figure 3-10. At discrete intervals, transfers of the upper phase take place from one vessel to the next. Among these transfer steps, the upper phase then present in each vessel is equilibrated with the lower phase in that vessel. A small amount of feed mixture is initially present in the first vessel and then carried along from vessel to vessel in the distribution process. For a component *A* being separated, the formulation is presented in Example 3-5.

Figure 3-10 Craig countercurrent distribution.



EXAMPLE 3-5 A chromatographic separation.

$$M_{A_{0,0}} = M_{A_F}$$

$$M_{A_{p,0}} = 0 \qquad \forall p \in 1 \dots N$$

$$\begin{bmatrix} K_{A_{p,t}} = K_A \\ \left[\begin{matrix} \dfrac{M_{A_{p,t}}}{M_{A_F}} = \dfrac{t!}{t! \cdot (t-p)!} \cdot f_{A_{p,t}}^{p} \cdot (1 - f_{A_{p,t}})^{(t-p)} \\ t \geq p \end{matrix} \right] \vee \left[\begin{matrix} \dfrac{M_{A_{p,t}}}{M_{A_F}} = 0 \\ t < p \end{matrix} \right] \\ M_{A_{p,t}} \leq M_{A_C} \end{bmatrix} \vee \begin{bmatrix} K_{A_{p,t}} = K_A' \cdot \left(\dfrac{M_{A_{p,t}}}{M_{A_F}} \right)^{0.05} \\ M_{A_{p,t}} = f_{A_{p,t}} \cdot M_{A_{p-1,t-1}} + (1 - f_{A_{p,t}}) \cdot M_{A_{p,t-1}} \\ M_{A_{p,t}} > M_{A_C} \end{bmatrix}$$

$$f_{A_{p,t}} = \frac{K_{A_{p,t}} \cdot (V_U / V_L)}{1 + K_{A_{p,t}} \cdot (V_U / V_L)}$$

$$\forall p \in 1 \dots N, \forall t \in 1 \dots T$$

In Example 3-5, V_U and V_L are the volumes of the upper and lower phase correspondingly, $f_{A_{p,t}}$ is the fraction of A present in the upper phase in the vessel *p* after the transfer step *t*, $M_{A_{p,t}}$ is the total amount of *A* in the vessel p after the transfer step *t*, $K_{A_{p,t}}$ is the equilibration ratio of *A* in the vessel p after the transfer step *t*. $K_{A_{p,t}}$ can be a constant, but it also can be a function of the total amount $M_{A_{p,t}}$. M_{A_F} is the initial amount of *A* in the separation process. Finally, K_A and K_A' are given constants depending on the component *A*. M_{A_F} and M_{A_C} are always assumed as given. A consistency analysis of a representative structure shows that, by defining only V_U and V_L as the set of independent variables, the system is structurally consistent for any number of vessels *p* and transfer steps *t*.

3.6 SUMMARY

In this chapter, we have reviewed the concepts involved in structural analysis. We then derived an extension to Zaher's consistency analysis of conditional models. This extension allows a consistency analysis to be applied to conditional models in which the number of variables and equations for each of the alternatives may not be the same. In general, we think that, in order to ensure the structural consistency of a conditional model, the combinatorial consistency analysis must be performed. However, we have shown that, by taking advantage of the structure of the problem, it is sometimes possible to reduce the computational effort required by the consistency analysis. In addition, we used simple examples to illustrate the relevant definitions in structural analysis, to demonstrate the scope of application of the extension to Zaher's consistency analysis, and to show how we can take advantage of the existence of common incidence patterns and repeated structures in the structural analysis of conditional models.

3.7 REFERENCES

Allan, B. A.; A More Reusable Modeling System; Ph.D. thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA,1997.

Allen, G. L. and Westerberg, A. W.; Solution Procedures for Indexed Equation Sets: Structural Considerations. *AIChE Journal*, 22(3):549–558, May 1976.

Barton, P. I.; The Modeling and Simulation of Combined Discrete/Continuous Processes. Ph.D. thesis, Department of Chemical Engineering, Imperial College of Science, Technology and Medicine, 1992.

Duff, I. S., Erisman, A. M. and Reid, J. K.; Direct Methods for Sparse Matrices. Monographs on Numerical Analysis, Oxford Science Publications, Oxford University Press, New York, 1989.

King, C. J.; Separation Processes, Chemical Engineering Series, 2nd. Edition, McGrawHill, 376-382, 1980.

Piela, P.; ASCEND: An Object-Oriented Computer Environment for Modeling and Analysis. Ph.D. thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, 1989.

Westerberg, A. W., Hutchinson, H. P., Motard, R. L. and Winter, P.; Process Flowsheeting. Cambridge University Press, 1979.

Zaher, J. J.; Conditional Modeling. Ph. D. thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, 1995.

Zaher, J. J.; Conditional Programming. The Annual AIChE National Meeting, March 1993.

CHAPTER 4 A BOUNDARY CROSSING ALGORITHM

In this chapter, we investigate the solving of conditional models using a boundary crossing algorithm proposed by Zaher (1995). This algorithm involves the execution of several well differentiated activities including logical analysis, continuous reconfiguration of the equations constituting the problem, calculation of Newton-like steps, and the calculation of gradient steps. We describe the practical implementation of this boundary crossing algorithm as a conditional modeling solution tool. In such an implementation, we have integrated the entities created and/or used to perform each of the activities in an object-oriented solving engine: the conditional modeling solver CMSlv. Also, we describe and solve several examples of conditional models in chemical engineering. Finally we discuss the scope and limitations of the algorithm.

4.1 BACKGROUND

While dealing with conventional models, one generally applies iterative nonlinear techniques in order to solve the equations simultaneously. Newton-like numerical techniques are the methods most commonly applied to find the solution of conventional models. Such methods require both continuity and differentiability of the system of equations over the entire feasible region in order to guarantee convergence.

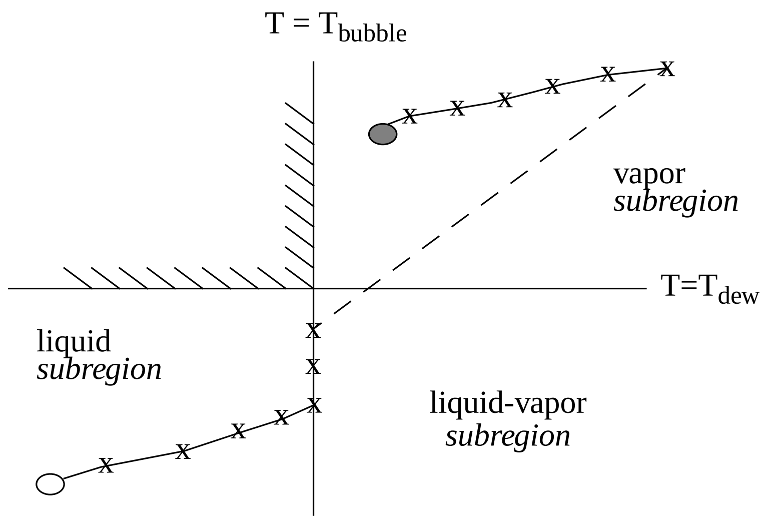
In conditional models, however, the equations within each alternative set are confined to a subset of the search space, which results in the dissection of the feasible region into subregions (Zaher, 1995). Because of the loss of continuous functionality between the surfaces of neighboring subregions in the search space, Newton-like iterative techniques cannot be applied if the solution to the system requires a step from one subregion to another.

The literature reports several approaches to the solution of conditional models. One can formulate conditional models as mixed integer programming problems. Grossmann and Turkay (1996) show that, in the case of linear equations, the solution of a conditional model can be found by solving a MILP problem that is often solvable as a relaxed LP. Also, based on the works of Raman and Grossmann (1993, 1994), Turkay and Grossmann (1996) address the solution of disjunctive set of nonlinear programming problems. They show that the use of logic can improve the efficiency and robustness of MINLP algorithms in the solution of structural optimization problems. Zaher (1995) makes the following observations about the MINLP approach (big-M formulation) to the solution of conditional models:

1. One can expect that, during the solution of a MINLP, the algorithm will jump from one subregion to another which is remote from it, leading to the poor initialization of some of the nonlinear subproblems (See Figure 4-1).
2. The nonlinear subproblems insist on converging the equations of what might be an incorrect subregion.
3. The MINLP remains as a rigorous, general purpose algorithm for problems where discrete decision making cannot be avoided.

Besides the mixed-integer formulation, some alternative approaches exist which, in principle, can avoid the need of discrete decisions. This chapter focuses on the implementation of one such approach, a boundary crossing algorithm (Zaher, 1991).

Figure 4-1 MINLP approach to the solution of conditional models. Note the jump from the liquid to the vapor region, leading to poor initial conditions from which to converge the vapor model equations.



4.2 THE BOUNDARY CROSSING ALGORITHM

Zaher (1991,1995) introduced a boundary crossing algorithm as an alternative approach to the solution of conditional models.

4.2.1 THE FORMULATION

In conditional models, the equations within each alternative set are confined to a subset of the feasible region. In the boundary crossing algorithm, a set of boundary expressions of the form $\underline{b}(\underline{x}) \geq 0$ is used to associate each set of conditional equations to each of the subregions. Each boundary expression partitions the feasible region into two neighboring subregions, each of the subregions characterized by either the satisfaction or non-satisfaction of the expression. Thus, each subregion in the feasible region is identified by a unique combination of truth values for all of the boundary expressions. In this way, the partitioning of the feasible region into subregions is combinatorial. If l is the number of boundary expression (i.e. $b_i(\underline{x}) \geq 0$ where $i \in \{1 \ldots l\}$), the number of subregions is 2^l. For most of the conditional models many of the subregions are meaningless. These subregions, therefore, should be removed from the feasible region and no conditional equations need to be associated to them. One must emphasize that the boundary expressions are not inequality constraints but rather are used to associate values of the modeling variables with the appropriate set of equations. A way of identifying a specific subregion is by using a set s where $s \in \{1 \ldots l\}$. The set s contains the indices of the boundary expressions which must be satisfied in order for the solution to reside in that specific subregion. If R is the set of all of the subsets s belonging to the feasible region with which m conditional equations are to be associated in the form $\underline{r}_s(\underline{x}) = 0$, then the conditional model can be formulated in the following disjunctive representation:

$$\underset{s \in R}{\vee} \begin{bmatrix} b_i(\underline{x}) \geq 0, \forall i \in s \\ b_i(\underline{x}) < 0, \forall i \in \{1 \ldots l\} - s \\ \underline{r}_s(\underline{x}) = 0 \end{bmatrix} \tag{4.1}$$

Note that, when the number of boundary expressions is zero, only one subregion is created using $s = \varnothing$ and the previous formulation reduces to a conventional nonlinear system of equations.

Since frequently a large number of equations are common to all of the subregions, it is convenient to include those invariant equations outside the disjunction:

$$\underline{g}(\underline{x}) = 0$$

$$\underset{s \in R}{\vee} \begin{bmatrix} b_i(\underline{x}) \geq 0, \forall i \in s \\ b_i(\underline{x}) < 0, \forall i \in \{1 \ldots l\} - s \\ \underline{r}_s(\underline{x}) = 0 \end{bmatrix} \tag{4.2}$$

where now $\underline{g}(\underline{x})$ is m_g dimensional and $\underline{r}_s(\underline{x})$ is m_r dimensional such that $m_g + m_r = m$.

Note that the formulations given by (4.1) and (4.2) represent one disjunction over all of the subregions in the feasible region of a conditional model. Such a global disjunctive statement can be generated from the combination of all of the individual disjunctions in the problem. However, it is not necessary to formulate the conditional model strictly in the form given by (4.1) or (4.2) in order to solve it by using the boundary crossing algorithm. As we describe in the following section, the boundary crossing algorithm considers only the subregion in which the current point lies (if the current point lies inside a subregion) or only the subregions in the neighborhood of a boundary (if the current point lies at a boundary). In that way, a representation in which each individual disjunction is explicitly formulated is also suitable. Such a representation is essentially the same as in Equation (1.1). Note, however, that the domain of validity of the conditional equations can be expressed in terms of a set of boundary expressions, as we do, instead of a set of inequality constraints as given by Grossmann and Turkay (1996) in formulation (1.1).

4.2.2 THE SOLUTION ALGORITHM

The goal of a conditional model solver is to find a solution to the disjunctive system of equations (4.2) (or to the system of equations (1.1)), a solution which consists in a vector of variables satisfying the invariant set of equations and exactly one set of conditional equations, providing that the truth values of the boundary expressions are consistent with the set of variant equations constituting the solution.

The most popular iterative numerical techniques for the solution of a system of nonlinear equations involve strategies designed to enforce descent of some objective function with each iteration. This objective function is generally chosen as the square norm of the residuals of the model equations, assuming the equations are all appropriately scaled. Hence, for each subregion *s*, the objective function $\phi_s(\underline{x})$ is defined by:

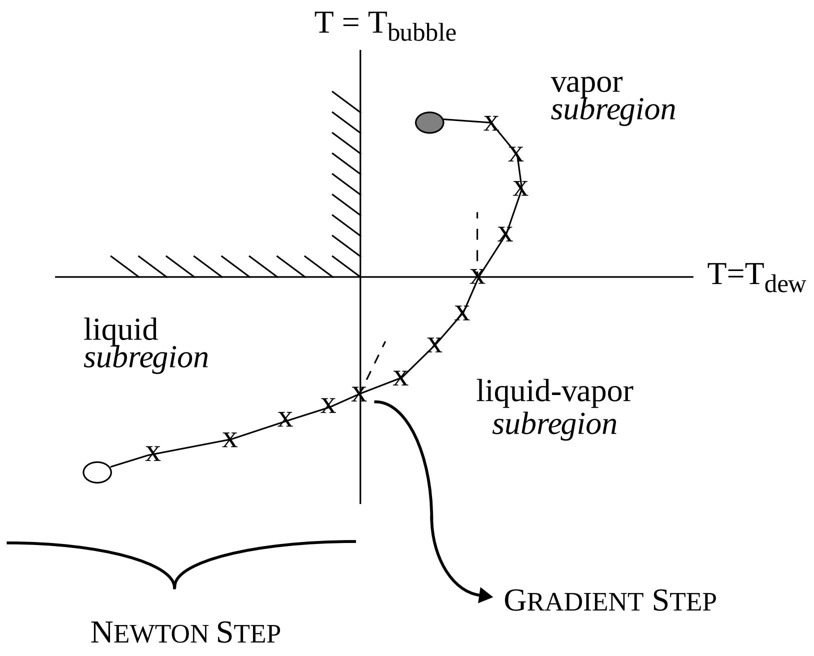
$$\underline{f}_s(\underline{x}) = \begin{bmatrix} \underline{g}(\underline{x}) \\ \underline{r}_s(\underline{x}) \end{bmatrix}$$

$$\phi_s(\underline{x}) = \frac{1}{2} \cdot \underline{f}_s^T(\underline{x}) \cdot \underline{f}_s(\underline{x}) \qquad \textbf{(4.3)}$$

In the boundary crossing algorithm, all of the points where the functionality is nondifferentiable are characterized by boundary hyper-planes as given by the boundary expressions $\underline{b}(\underline{x}) \geq 0$. Therefore, the equations within each subregion form a continuous and differentiable objective function. Hence, a Newton-based numerical technique may find the solution when the initial point is internal to the correct subregion. On the other hand, if one places the initial point inside an incorrect subregion, one could expect to encounter a subregion boundary in a finite number of iterations.

According to the above discussion, the boundary crossing algorithm applies a Newton-like step for all the points internal to a subregion, taking advantage of the superlinear convergence feature of such a step. If during the iterative process a boundary is reached, then a Newton-like step cannot be taken because of the nondifferentiability of the objective function at that point. A boundary crossing based on gradient methods (using linear order convergence methods) takes place instead. Once off the boundary, the superlinear convergence steps resume possibly using a new set of equations. Figure 4-2 illustrates the boundary crossing algorithm.

Figure 4-2 The boundary crossing algorithm in a simple flash equilibrium calculation.



4.2.2.1 Boundary Crossing

In the boundary crossing algorithm, because of the lack of differentiability at a boundary, an analogue of the steepest descent vector is derived in order to advance the solution progress into the correct subregion.

While solving continuous and differentiable models, the steepest descent direction at some point with values $\underline{a}$ for the variables is in the opposite direction of the gradient of the objective function at that point. Hence, a vector $\underline{d}$ with a direction of steepest descent direction is a vector which solves the following problem:

$$\begin{aligned} &\min \quad \nabla_x \phi(\underline{a})^T \cdot \underline{d} \\ &\text{s.t.} \quad \underline{d}^T \cdot \underline{d} \le 1 \end{aligned} \tag{4.4}$$

When a descent direction exists at $\underline{a}$, the solution vector $\underline{d}$ has the same direction as $-\nabla_x \phi(\underline{a})$ but it is of unit length. In other words, the steepest descent direction for a differentiable function is the direction which yields a minimum in the directional derivative.

For a conditional model, each subregion has a different value for the directional derivative at a boundary. For that case, Zaher (1991) defines the directional derivative as the maximum over all of the directional derivatives. If B is the set of all of the subregions in the neighborhood of a boundary, then the directional derivative for the problem at such boundary is given by:

$$\begin{aligned} &\max \quad \nabla_x \phi_i(\underline{a})^T \cdot \underline{d} \\ &\text{s.t.} \quad i \in B \end{aligned} \tag{4.5}$$

Hence, the minimum of this directional derivative constitutes the steepest descent direction at the boundary:

$$\begin{aligned} &\min \quad \begin{pmatrix} \max & \nabla_x \phi_i(\underline{a})^T \cdot \underline{d} \\ \text{s.t.} & i \in B \end{pmatrix} \\ &\text{s.t.} \quad \underline{d}^T \cdot \underline{d} \le 1 \end{aligned} \tag{4.6}$$

The definition of directional derivative given by (4.5) aims for a descent direction with respect to all of the subregions neighboring the boundary. By doing that, one will prevent a cycling situation in which a loop of connected subregions are visited continually.

The problem given by (4.6) is recognized as a mini-max problem. Because of the convexity of this problem, one can derive the dual problem to obtain:

$$\begin{array}{rl} min & \underline{d}^T \cdot \underline{d} \\ s.t. & \underline{d} = -N \cdot \underline{\alpha} \\ & \underline{e}^T \cdot \underline{\alpha} = 1 \\ & \alpha_i \geq 0, \forall i \in B \end{array} \qquad \textbf{(4.7)}$$

where $\underline{e}$ is a vector with all its elements equal to one and N is a matrix whose columns are the vectors of gradients of the subregions in the neighborhood of the current boundary. That is, column i of N is the vector $\nabla_x \phi_i(\underline{a}), \forall i \in B$. Also, both $\underline{e}$ and $\underline{\alpha}$ are vectors with dimension equal to the number of subregions neighboring the current boundary. The derivation of equation (4.7) from (4.6) has been described by Zaher (1991,1995) and it is given in Appendix B. If the gradients of all of the neighboring subregions cannot be confined to one side of the boundary hyper-plane, the result of (4.7) is a minimum of zero and we terminate the algorithm. We terminate because no descent direction with respect to all the neighboring subregions has been found. Otherwise, If the result of (4.7) is nonzero, the linear combination of the elements of the vector $\underline{\alpha}$ is used to generate the steepest descent direction, along of which one should move away from the boundary.

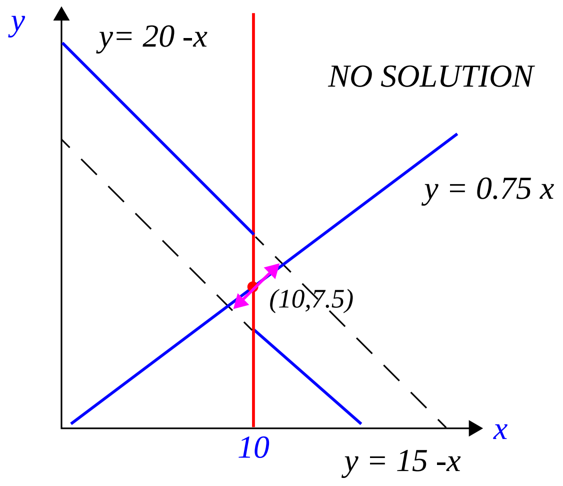
4.2.2.2 Assumption of Continuity in the Boundary Crossing Algorithm

In the boundary crossing algorithm, one has to assume the continuity of the conditional model. In conditional models, continuity is ensured as follows: if the solution to the equations of an arbitrary subregion lies on a boundary hyperplane, then that solution must satisfy the equations of all of the subregions neighboring that boundary.

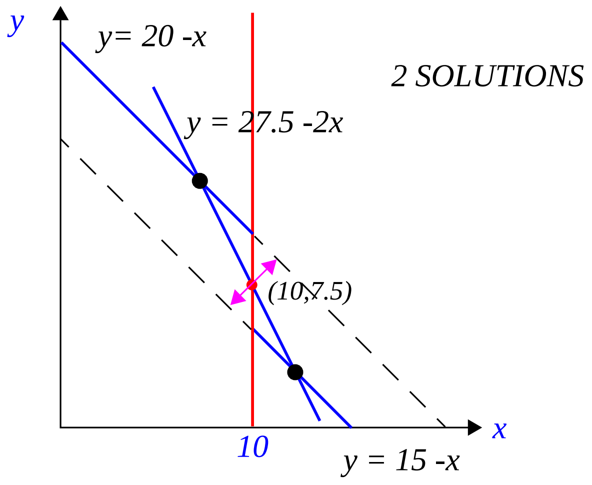
The reason for this assumption of continuity is best described by analyzing Figure 4-3. Figure 4-3 presents two cases in which continuity is not preserved in the conditional model and discovers the potential difficulties of such cases. In case (a) there is no solution to the problem, and in case (b) the conditional model has two solutions. Note that, in both cases in Figure 4-3, the vectors of gradients of the subregions neighboring the boundary are of opposite direction at the point *(x=10,y=7.5)*. If we solve problem (4.7) for such a point at the boundary, the solution of (4.7) would be zero in both cases, and the boundary

Figure 4-3 Vectors of gradients are of opposite direction at a boundary.

(a)



(b)



crossing algorithm would terminate. Hence, for the case (b), the boundary crossing algorithm would terminate without finding any of the two existing solutions to the conditional model. A possible cure for this complication would be to search in each neighboring subregion for a solution interior to any of them. Such an approach has been considered throughout this research, but it is not yet a part of the implementation that we describe in this Chapter.

4.3 IMPLEMENTATION IN AN EQUATION BASED ENVIRONMENT

In this section, we present some of the details of the implementation of the boundary crossing algorithm in an equation-based environment.

In Figure 4-4, we show a flowsheet of the main activities that a solver applying the boundary crossing algorithm must perform. Given the initial guess for the model variables, the algorithm starts by evaluating the logical boundary expressions in order to determine whether that initial guess resides on a boundary hyper plane or it is internal to any of the subregions:

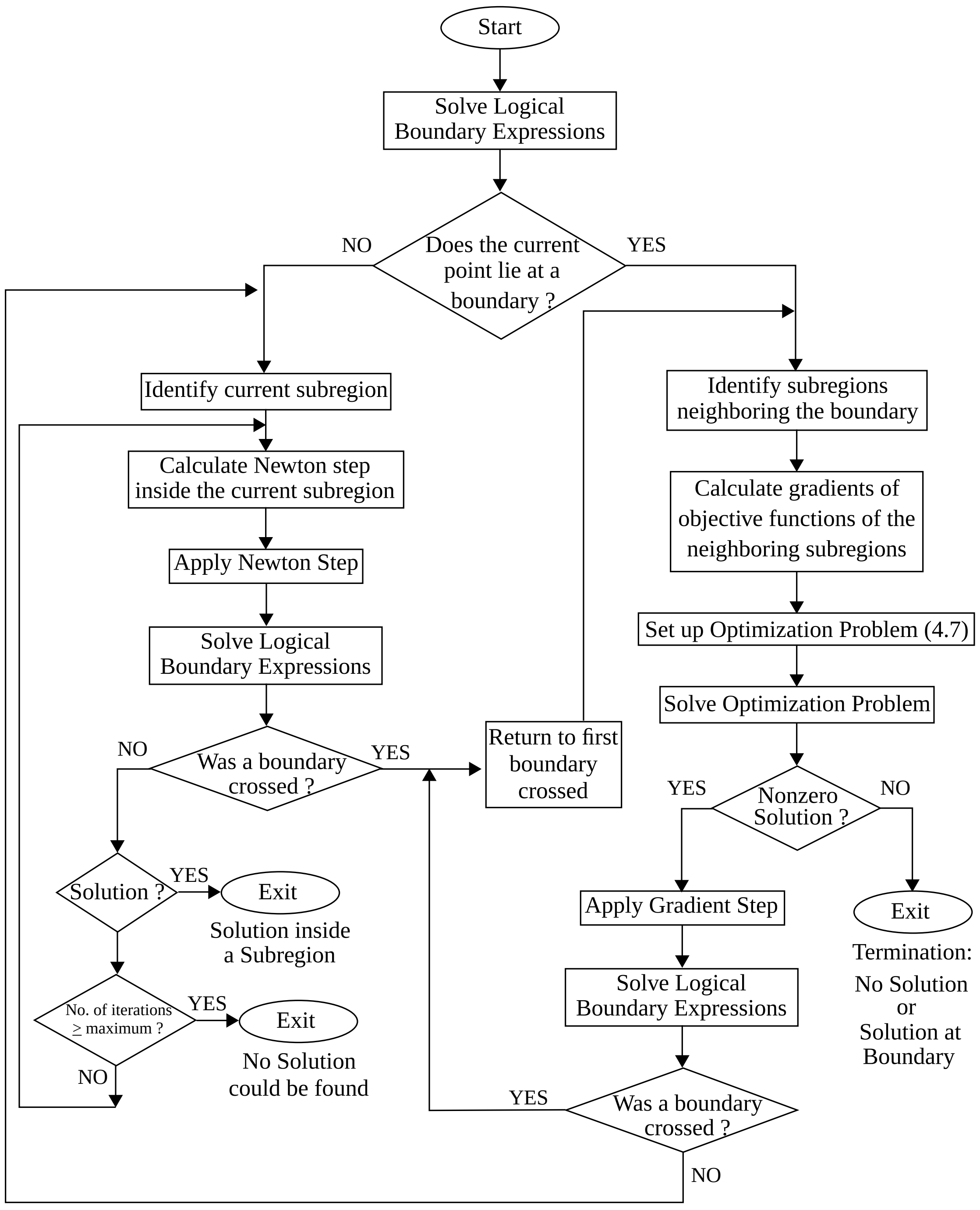
- If the point resides on a boundary, a gradient step has to be calculated. In order to do that, the equations and variables of all of the subregions neighboring the boundary have to be identified and differentiated. The problem given by (4.7) is then generated and solved to obtain the steepest descent direction. If the result of (4.7) is a minimum of zero, then the algorithm terminates concluding that no descent direction could be found for the current point on the boundary. If the solution of (4.7) is nonzero, then the gradient step is taken. After this step, it is necessary to verify if one or several boundaries have been crossed. If so, the gradient step is reduced until the problem lies exactly on the first boundary crossed, and a gradient step must be calculated for the new boundary. If no boundary has been crossed, the algorithm proceeds to identify the variables and equations of the current subregion and a nonlinear conventional technique is the used to move towards the solution as explained below.
- If the point does not reside on a boundary, it is necessary to identify the current

subregion. That means that we need to identify the equations and variables constituting the problem. The algorithm assumes that the starting point corresponds to a subregion for which the system of equations are defined, i.e. the initial subregion is feasible. An iterative technique is then used in order to generate a Newton-like step towards the solution. After each step, one needs to verify if some of the boundaries have been crossed. If no boundary has been crossed, the numerical technique continues the iterative process in the current subregion until the convergence criteria is satisfied (termination of the algorithm) or until a boundary is encountered. When a boundary has been crossed, the length of the Newton-like step has to be reduced until the system resides exactly on the boundary, and then a boundary crossing has to be performed by using a gradient step as explained above.

An efficient implementation of this algorithm in an equation-based environment is hard for the following reasons:

1. The implementation requires the incorporation of modeling tools which enables the user of an equation-based environment to represent conditional models. Mainly, these modeling tools have to be capable of representing alternative sets of equations and the logical conditions given by the boundary expressions.
2. The boundary crossing algorithm involves the performance of several different tasks, including logical analysis, periodic reconfiguration of the equations and variables of the system (switching among alternative subregions), performance of a Newton-like step (which requires the incorporation of a conventional nonlinear technique) and performance of a gradient step (which requires the solution of an optimization problem). Thus, in the implementation of this technique one has to provide the numerical and algorithmic tools needed to execute each of the previous tasks and to integrate them in a procedural solving engine.

Figure 4-4 Flowsheet of the boundary crossing algorithm implementation.


Start
Solve Logical Boundary Expressions
Does the current point lie at a boundary ?
NO
YES
Identify current subregion
Calculate Newton step inside the current subregion
Apply Newton Step
Solve Logical Boundary Expressions
Was a boundary crossed ?
NO
YES
Solution ?
YES
Exit
Solution inside a Subregion
No. of iterations ≥ maximum ?
YES
Exit
No Solution could be found
NO
Return to first boundary crossed
Identify subregions neighboring the boundary
Calculate gradients of objective functions of the neighboring subregions
Set up Optimization Problem (4.7)
Solve Optimization Problem
Nonzero Solution ?
YES
NO
Apply Gradient Step
Exit
Termination: No Solution or Solution at Boundary
Solve Logical Boundary Expressions
Was a boundary crossed ?
YES
NO

4.3.1 THE MODELING TOOLS

We use the modeling tools described in Chapter 2, which allow the representation of conditional models in an equation based environment Those modeling capabilities meet the representation needs of the boundary crossing algorithm:

1. Boundary expressions can be represented by combining the CONDITION statement and logical relations.
2. Alternative sets of model equations can be represented by using the WHEN statement.

4.3.2 THE SOLVING ENGINE

We implement our conditional modeling solver in a server-client, object-oriented architecture. Figure 4-5 describes this implementation. This conditional modeling solver (CMSlv) is in charge of the reconfiguration of the appropriate sets of equations and variables at each step of the algorithm, convergence tests, and the generation of the optimization subproblem when the system resides on a boundary. In addition, the conditional modeling solver is also in charge of the management of external entities. Such external entities are in charge of executing some of the specific tasks that the algorithm requires. Particularly, calls to external entities are done for the execution of:

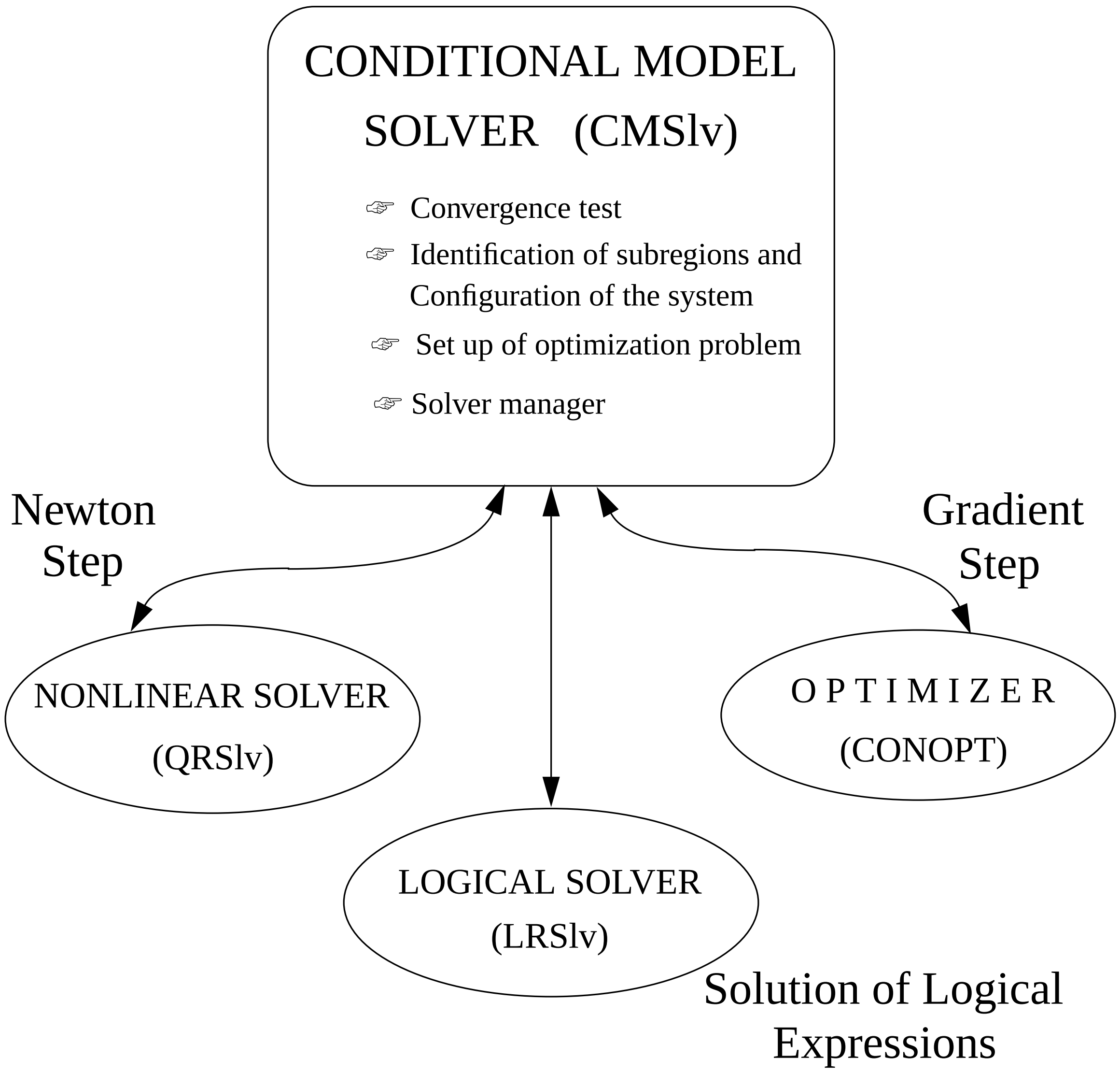
1. The analysis of logical boundary expressions.
2. The calculation of a step inside a subregion (Newton-like superlinear iterative technique).
3. The calculation of a gradient step by solving an optimization subproblem.

In the remainder of this section, we will describe in more detail each of these tasks as well as the external entities we created or used to provide support to the conditional solver CMSlv.

4.3.2.1 Logical Analysis

The domain of validity of the alternative sets of equations in the boundary crossing algorithm is given in terms of the truth value of a set of logical boundary expressions. Hence, logical expressions have to be constantly evaluated during the iterative solution scheme. In order to perform this task, we created the logical solver LRSlv.

Figure 4-5 Object-oriented architecture of the boundary crossing implementation.

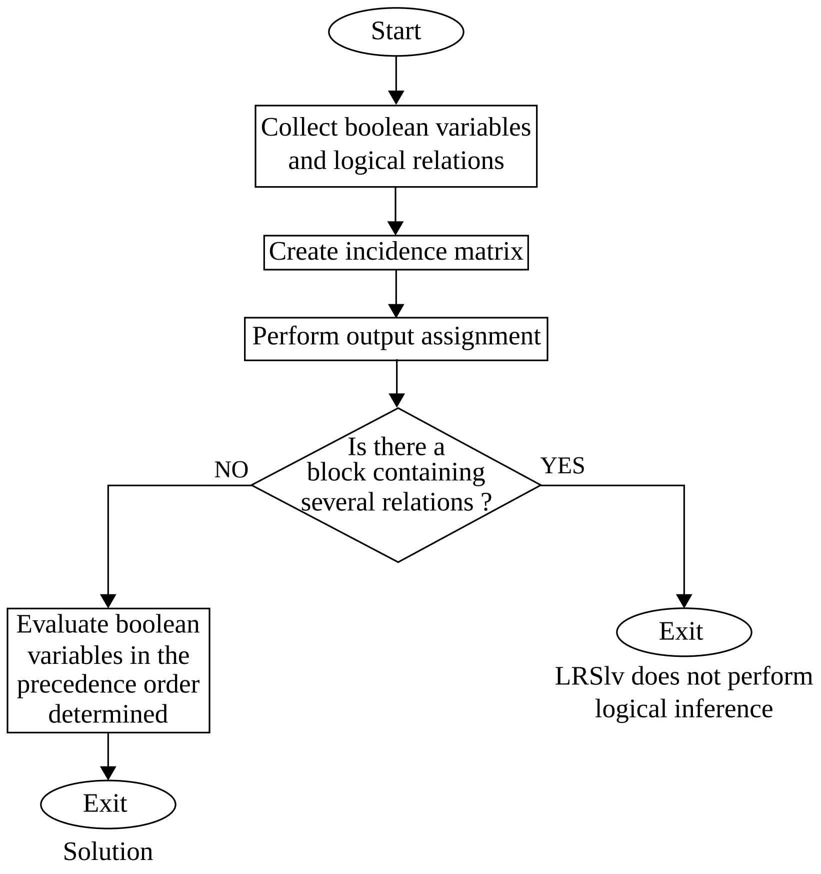

CONDITIONAL MODEL SOLVER (CMSlv)
☞ Convergence test
☞ Identification of subregions and Configuration of the system
☞ Set up of optimization problem
☞ Solver manager
Newton Step
Gradient Step
NONLINEAR SOLVER (QRSlv)
OPTIMIZER (CONOPT)
LOGICAL SOLVER (LRSlv)
Solution of Logical Expressions

LRSlv has been incorporated into the ASCEND modeling environment. The ASCEND modeling language allows a logical boundary expression to be represented in terms of a combination of:

- Boolean variables.
- Boolean operators.
- The truth value of conditions expressed in term of real variables.

Hence, the logical solver LRSlv has been implemented in such a way that it is able to handle this type of flexibility. Figure 4-6 shows the flowsheet of the activities performed by LRSlv. In this implementation, we assume that we can explicitly evaluate all the boolean variables in some precedence order. In other words, LRSlv does not perform logical inference. Therefore, while finding the value of a boolean variable, we require the truth values of all the boolean variables and conditions on which each boolean variable depends to have been previously given or previously evaluated. Also, we assume that we can evaluate at any time the truth value of a condition expressed in terms of real variables. Both of our assumptions are met when the boundary crossing algorithm is being used to solve conditional models. LRSlv has also been attached as an external entity providing support to CMSlv as explained above.

Figure 4-6 Implementation of the logical solver LRSlv.


Start
Collect boolean variables and logical relations
Create incidence matrix
Perform output assignment
Is there a block containing several relations ?
NO
YES
Evaluate boolean variables in the precedence order determined
Exit
Solution
Exit
LRSlv does not perform logical inference

4.3.2.2 Configuration of the System of Equations

The conditional solver CMSlv is capable of determining whether the current point resides on a boundary or is internal to any of the subregions. If the point is internal to any of the subregions, CMSlv must identify the equations and variables constituting the problem in that specific subregion. To perform this task, CMSlv uses the notion of an active equation and an active variable. By active we mean "it is part of the problem currently being solved." Computationally speaking, to set a relation as active or inactive implies a simple bit operation. A consistent mechanism to select the structure of the system inside a specific subregion was described in Chapter 2. That mechanism is used in this implementation.

4.3.2.3 Conventional Nonlinear Solver: Newton-like Step

When the values of the variables of the model correspond to a point internal to any of the subregions, the boundary crossing algorithm requires the calculation of a Newton-like step. CMSlv calls an external solver to perform this task. We do not impose any restrictions for the selection of this external solver. CMSlv can be able to interact with any solver which is able to calculate a step based on derivative and function evaluations and on the structural information of the system of equations. In the current implementation of CMSlv, the Newton-like step is calculated by the ASCEND nonlinear solver QRSlv (Westerberg, 1989). In order to enforce the descent of the objective function local to the subregion, QRSlv uses the modification to the Levenberg-Marquardt algorithm given by Westerberg and Director (1979).

4.3.2.4 Setting Up the Optimization Subproblem

When CMSlv determines that the current point of the system resides on a boundary, a gradient step has to be calculated by solving an optimization subproblem. Setting up this optimization subproblem requires the calculation of the gradients of the objective functions of all of the subregions neighboring a boundary. It results in a system of equations which is quite different from the system of equations of any of the subregions of the original problem. That implies that the data structures required to provide the information about this optimization subproblem have to be dynamically created and destroyed. In addition, in order to minimize the number of function and gradient evaluations employed for setting up each optimization subproblem, it is necessary to identify the invariant set of equations (if there is one) and to avoid multiple evaluations for those equations. Figure 4-7 presents the flowsheet of the activities performed by CMSlv while setting up an optimization subproblem at a boundary.

4.3.2.5 Solution of Optimization Subproblems: Gradient Step

After CMSlv has created the optimization subproblem at a boundary, this subproblem is solved with a call to another external entity. This time, the external solver must be an optimizer. Once again, there are no limitations in the implementation of CMSlv as to which optimizer should be used. The current implementation interacts with the subroutine CONOPT (Drud, 1985), which uses a reduced gradient approach to the solution of the optimization problem. CMSlv provides the values of the derivatives of the linear constraints (elements of the vectors of gradients of the objective functions), the residual of these constraints, and the derivatives and values of the objective function. CONOPT (like QRSlv in the case of the calculation of the Newton-like step) is considered a black box which provides the gradient step based on that information.

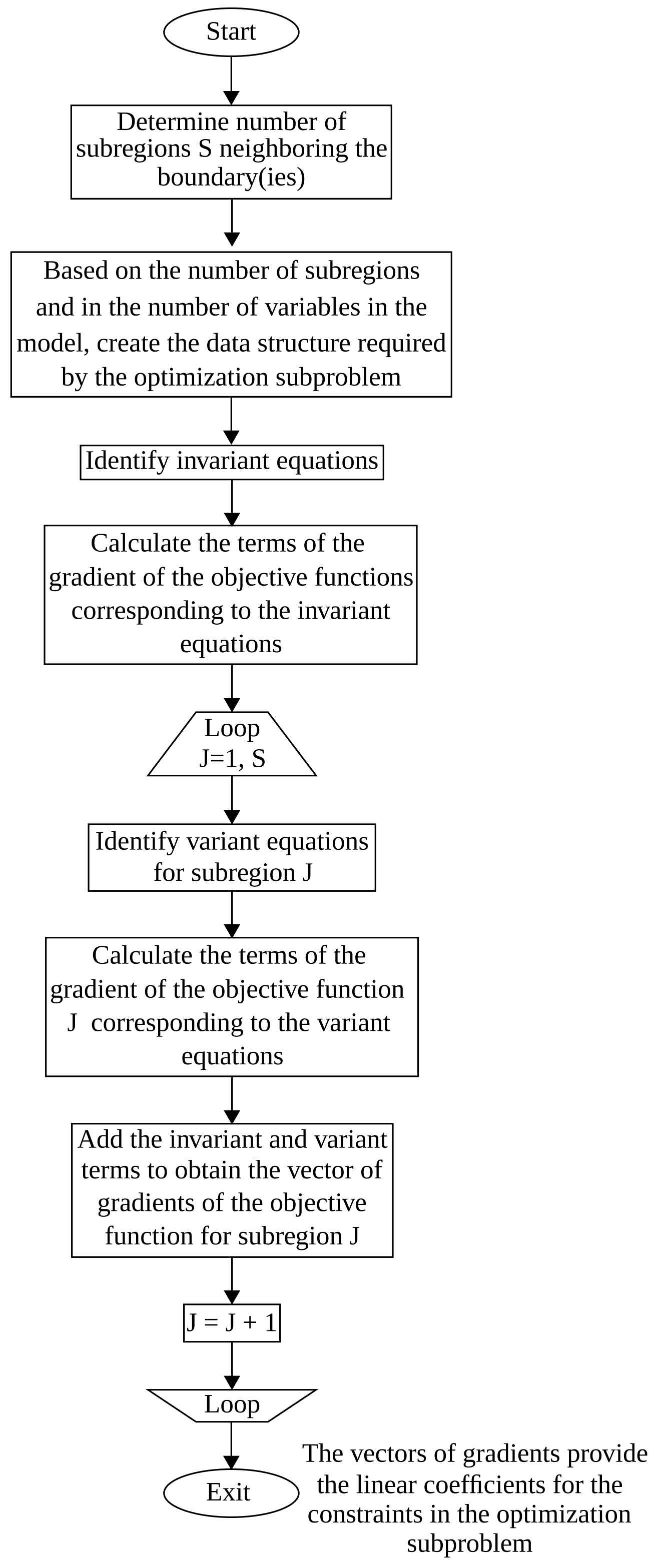
4.3.2.6 Termination

Termination of the algorithm can occur on a point internal to any subregion or on a point residing on a boundary. It occurs on a boundary when the result of the optimization subproblem is zero, or occurs internal to a subregion when the norm of the residuals of the equations in that subregion is less than a specified tolerance. CMSlv is in charge of the convergence test after both the gradient step and the Newton-like step.

4.3.2.7 About CMSlv

Besides tasks like the reconfiguration of the system, termination tests, setting up of the optimization problem, etc., CMSlv implements the logic of the algorithm and interacts with the external solvers. It provides the information required by the external solvers (LRSLv, QRSlv and CONOPT) and processes the results obtained from the calls to them. The conditional modeling solver (CMSlv) and the logical analyzer (LRSlv) have been attached to the ASCEND environment.

Figure 4-7 Setting up the optimization subproblem.


Start
Determine number of subregions S neighboring the boundary(ies)
Based on the number of subregions and in the number of variables in the model, create the data structure required by the optimization subproblem
Identify invariant equations
Calculate the terms of the gradient of the objective functions corresponding to the invariant equations
Loop J=1, S
Identify variant equations for subregion J
Calculate the terms of the gradient of the objective function J corresponding to the variant equations
Add the invariant and variant terms to obtain the vector of gradients of the objective function for subregion J
J = J + 1
Loop
Exit
The vectors of gradients provide the linear coefficients for the constraints in the optimization subproblem

4.4 ILLUSTRATIVE EXAMPLES

We have modeled and solved several examples of conditional models found in the literature by using the ASCEND modeling language and our implementation of the boundary crossing algorithm. Here we give a detailed description of each of these examples. Also, in Appendix C we show a representative section of the ASCEND model for each of the examples.

EXAMPLE 4-1 Fluid transition (Zaher, 1995).

This example describes the flow of a compressible gas in an adiabatic frictional circular pipe of constant diameter. Nonsmooth functionality occurs due to the possible transition between sonic-subsonic flow at the outlet of the pipe. The alternatives for the solution of the problem are represented by:

$$\begin{bmatrix} P_d - P_f < M_f - 1 \\ M_f - 1 = 0 \end{bmatrix} \vee \begin{bmatrix} P_d - P_f \geq M_f - 1 \\ P_d - P_f = 0 \end{bmatrix}$$

in which one of the terms corresponds to sonic flow (Match number $M_f=1$) and the other to subsonic flow ($P_d=P_f$). The equations describing the thermodynamics are omitted for simplicity and can be found in Zaher (1995). This example corresponds to a simplest case of a conditional model, which involves only one boundary expression (and, therefore, only $2^1=2$ subregions) and contains only one conditional equation in each alternative set of variant equations.

EXAMPLE 4-2 Phase equilibria (Zaher, 1995).

An isothermal flash is applied to a ternary system involving benzene, ethanol and water. According to the phase diagram of this mixture and, depending on the values of pressure and temperature, three phases can be expected to exist simultaneously, an aqueous liquid phase, an organic liquid phase, and a vapor phase. The existence or nonexistence of each phase can be represented as a conditional statement. For instance, to represent the existence of the aqueous phase, the following statement applies:

$$\begin{bmatrix} \sum_{i \in C} y_{i_A} + \phi_A < 1 \\ \phi_A = 0 \end{bmatrix} \vee \begin{bmatrix} \sum_{i \in C} y_{i_A} + \phi_A \geq 1 \\ \sum_{i \in C} y_{i_A} = 1 \end{bmatrix}$$

as obtained by Michelsen (1982) and Zaher(1995). For a phase A, ϕ_A represents the fraction of the phase and $\underline{y}_A$ is a vector representing the compositions. Since there are three possible phases, we require three disjunctions to represent the behavior, which means that our search space contains $2^3=8$ subregions.

EXAMPLE 4-3 Heat exchanger (Zaher, 1995).

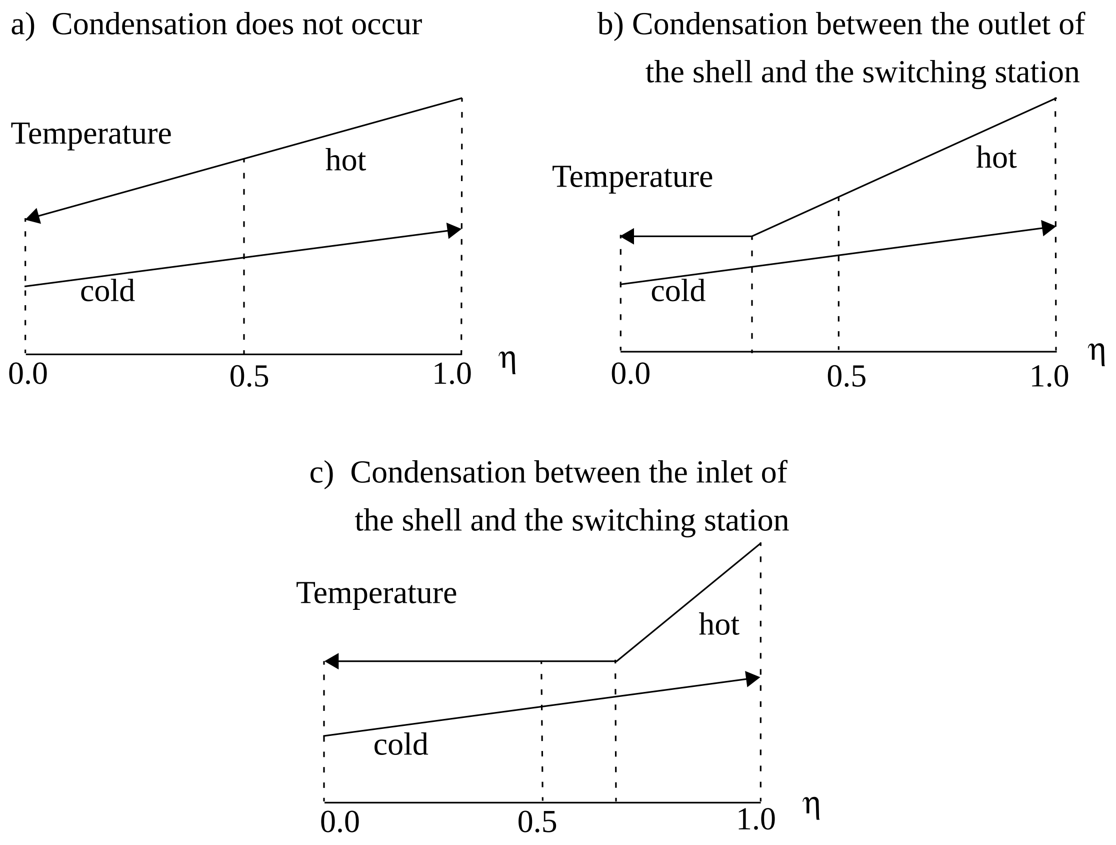
A very detailed explanation of this example can be found in Zaher (1995). It represents a case in which a conditional model contains differential equations that have to be integrated. The approach suggested is to discretize the differential equations and treat the problem as a conditional model with only algebraic equations. To accomplish this, Zaher (1995) introduced a "relay" method: the point in the domain of integration where transition occurs is continuously passed along, as a baton in a relay race, from one element to another by successive contractions and expansions of the individual elements. Switching stations at which the analogous baton transfer occurs must first be positioned. This example is introduced in Figure 4-8. Three finite elements are chosen with one switching station. To outline the three elements, four positions referenced by the indices {0...3} are used. The domain of integration is transformed to the dimensionless variable η which varies from zero to one. The difficulty with this model is that, in addition to solving for the temperature profile, the dimension of the finite elements are to be solved for as well.

Zaher (1995) shows that the three cases shown in Figure 4-8 can be represented as a conditional model including the following disjunctive statement:

$$\begin{bmatrix} \sum_{i \in C} x_{i_0} + \phi_0 < 1 \\ \sum_{i \in C} x_{i_2} + \phi_1 < 1 \\ \phi_0 = 0 \\ \eta_1 = 0 \\ \eta_2 = 0.5 \\ \phi_1 = 0 \end{bmatrix} \vee \begin{bmatrix} \sum_{i \in C} x_{i_0} + \phi_0 \geq 1 \\ \sum_{i \in C} x_{i_2} + \phi_1 < 1 \\ \sum_{i \in C} x_{i_0} = 1 \\ \sum_{i \in C} x_{i_1} = 1 \\ \eta_2 = 0.5 \\ \phi_1 = 0 \end{bmatrix} \vee \begin{bmatrix} \sum_{i \in C} x_{i_0} + \phi_0 \geq 1 \\ \sum_{i \in C} x_{i_2} + \phi_1 \geq 1 \\ \sum_{i \in C} x_{i_0} = 1 \\ \sum_{i \in C} x_{i_1} = 1 \\ \eta_1 = 0.5 \\ \sum_{i \in C} x_{i_2} = 1 \end{bmatrix}$$

where ϕ represents the fraction of the hot stream which is condensed and $\underline{x}$ is a vector representing the composition of the condensation droplets. As described above, the four

Figure 4-8 Alternative heat exchanger temperature profiles.


a) Condensation does not occur
Temperature
hot
cold
0.0
0.5
1.0
η
b) Condensation between the outlet of the shell and the switching station
Temperature
hot
cold
0.0
0.5
1.0
η
c) Condensation between the inlet of the shell and the switching station
Temperature
hot
cold
0.0
0.5
1.0
η

positions are referenced by the indices {0...3}. There are 2 boundary expressions in the model which would involve 4 subregions. However, one of the subregions is infeasible and can be eliminated from the search space.

EXAMPLE 4-4 Pipeline network (Bullard and Biegler, 1992).

Consider the pipe network shown in Figure 4-9 solved previously by Bullard and Biegler (1992). This problem can be described by the system of equations:

$$\sum_j Q_{ij} + \sum_j Q_{ji} = w_i \qquad \forall node\ i$$

$$H_{ij} = K \cdot sign(Q_{ij}) \cdot Q_{ij}{}^2 \qquad \forall arc\ ij\ without\ valve$$

$$\begin{bmatrix} K \cdot Q_{ij}{}^2 = 0 \\ H_{ij} \le 0 \end{bmatrix} \vee \begin{bmatrix} K \cdot Q_{ij}{}^2 = H_{ij} \\ H_{ij} \ge 0 \end{bmatrix} \qquad \forall arc\ ij\ with\ valve$$

$$H_{ij} = P_i - P_j \qquad \forall arc\ ij$$

$$Q_{ij} \ge 0 \qquad \forall arc\ ij\ with\ valve$$

The first equation is a flow balance around each node, the second is the Hazen-Williams relation for pipes with no valve, and the third is the relation between pressure drop and flowrate. Notice that an equivalent disjunctive representation for the Hazen-Williams relations can be given by:

$$\begin{bmatrix} H_{ij} = -K \cdot Q_{ij}{}^2 \\ Q_{ij} \le 0 \end{bmatrix} \vee \begin{bmatrix} H_{ij} = K \cdot Q_{ij}{}^2 \\ Q_{ij} \ge 0 \end{bmatrix} \qquad \forall arc\ ij\ without\ valve$$

All pipes are 100 ft long, 6 inches in diameter and with a roughness ε = 0.01 in, and the fluid is water: ρ = 62.4 lbm/ft3, μ = 1 cP. Pressures and inflow/outflow rates specifications are given in Table 4-1. Rates not specified are equal to zero (except the one in node 17 which is an unknown). Pressures not specified are unknowns in the problem. The starting point and converged flowrates are given in Table 4-2.

Figure 4-9 Pipeline network with five check valves.

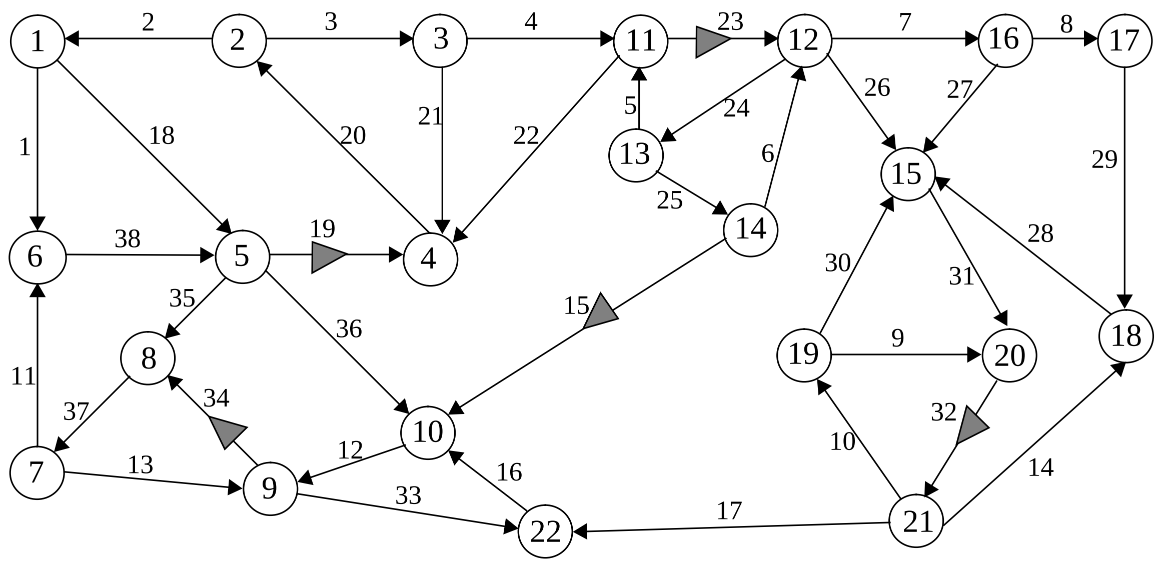


Table 4-1 Pressures and inflow/outflow rates for Example 4-4.

Node No.	Pressure P (psig)	Inflow rate w (gpm)
1		897.6
7		1570.9
11		-897.6
17	0	
20		-448.8
22		673.2

Since the problem contains 38 pipes, and, therefore, 38 boundaries are defined, the search space is constituted by 2^{38}=2.748779x10^{11} potential subregions. Hence, this example represents a case in which the combinatorial nature of the problem could have a severe effect in the performance of the boundary crossing algorithm.

Table 4-2 Starting point and converged flowrates for Example 4-4.

Pipe No.	Estimated flow $Q^{(0)}$ (gpm)	Converged flow Q* (gpm)
1	-48.5	-223.345
2	-640.7	-894.840
3	393.2	520.818
4	538.9	883.254
5	-32.3	-435.573
6	490.2	180.240
7	649.2	533.254
8	904.7	877.652
9	112.2	315.876
10	232.5	602.421
11	402.3	541.160
12	-420.4	-229.091
13	687.3	585.972
14	621.7	701.302
15	-719.2	0.0
16	52.7	-273.643
17	1199.0	-1303.723
18	305.4	226.105
19	582.8	943.137
20	-247.5	-374.022
21	-145.7	-362.435
22	-684.6	-954.723
23	293.6	504.804
24	-261.3	-255.153
25	-229.0	180.420
26	-395.1	407.123
27	-254.8	-344.398
28	-268.9	-216.346
29	-890.6	-917.648
30	120.3	286.546
31	-8.1	132.925
32	-344.7	0.0
33	473.1	356.88
34	-206.2	0.0
35	-275.0	-443.768
36	351.5	44.552
37	-481.2	-443.768
38	353.9	317.815

EXAMPLE 4-5 Simple L-V flash calculation (King, 1980).

This situation corresponds to a simple equilibrium calculation as given by King (1980). Basically, the problem consists of finding a solution to the well known Rachford-Rice equation:

$$x_i = \frac{z_i}{(K_i - 1) \cdot (V/F) + 1} \qquad y_i = \frac{K_i \cdot z_i}{(K_i - 1) \cdot (V/F) + 1}$$

$$\sum_i y_i - \sum_i x_i = 0 \quad \Longrightarrow \quad f(V/F) = \sum_i \frac{z_i \cdot (K_i - 1)}{(K_i - 1) \cdot (V/F) + 1} = 0$$

In the presence of two phases in equilibrium, the iterative calculation involves applying a convergence procedure until a value of *V/F* is found such that *f(V/F) = 0*. However, it

may well happen that the specifications of the problem do not correspond to a system with two phases present. For the case of a Liquid-Vapor equilibrium, King proposes the following criteria to differentiate among the different cases: *f(V/F)* will be positive at *V/F=0* and negative at *(V/F)=1*. Therefore, if *f(V/F)* is negative at *V/F=0*, the system is subcooled liquid. If *f(V/F)* is positive at *V/F=1*, the system is superheated vapor. This behavior can be represented in term of the following disjunctive statement:

$$\sum_i \frac{z_i \cdot (K_i - 1)}{(K_i - 1) \cdot (V/F) + 1} = R - V/F$$

$$\begin{bmatrix} V/F = 0 \\ R \le 0 \end{bmatrix} \vee \begin{bmatrix} V/F = R \\ 0 \le R \le 1 \end{bmatrix} \vee \begin{bmatrix} V/F = 1 \\ R \ge 1 \end{bmatrix}$$

For testing the proposed formulation, we took a mixture 20% of butane, 50% of pentane and 20 % of hexane, at 10 atm, and performed simulations for a broad range of temperatures (150 K to 890 K).

EXAMPLE 4-6 Linear mass balance (Grossmann and Turkay, 1996).

This example, illustrated in Figure 4-10, represents a problem in which each of the six processing units interconnected in a flowsheet contains three operating regions, each region with a different mass balance coefficient in terms of the main product flowrate. The mass balance coefficients and the bounds for each of the flowrates are shown in Table 4-3.

Because of the three operating regions, the disjunctive linear mass balance for each of the units is represented by a disjunction containing 3 disjunctive terms. For instance, for the case of unit operation 1, we have:

$$\begin{bmatrix} F_6 = 1.1 \cdot F_7 \\ F_{10} = 0.05 \cdot F_7 \\ 0 \le F_7 \le 50 \end{bmatrix} \vee \begin{bmatrix} F_6 = 1.15 \cdot F_7 \\ F_{10} = 0.1 \cdot F_7 \\ 50 \le F_7 \le 80 \end{bmatrix} \vee \begin{bmatrix} F_6 = 1.2 \cdot F_7 \\ F_{10} = 0.2 \cdot F_7 \\ 80 \le F_7 \le 150 \end{bmatrix}$$

The search space is constituted by 729 subregions. Also, the set of invariant equations as well as all of the sets of conditional equations contain only linear equations. Therefore, the equations in each subregion are all linear. The starting point used in this work and the converged values of the flowrates are shown in Table 4-4.

Figure 4-10 Processing units for Example 4-6.

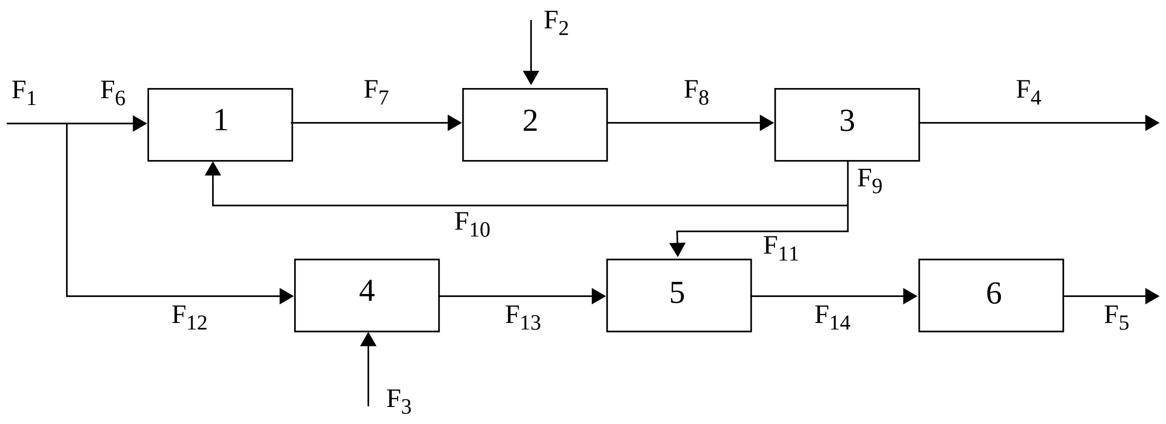


Table 4-3 Material balance equations for units in Example 4-6.

Unit	Main Product	Interval	Lower Bound	Upper Bound	Mass Balance Coefficient	
1	F_7	1	0	50	F_6: 1.10	F_{10}: 0.05
		2	50	80	1.15	0.10
		3	80	150	1.20	0.20
2	F_8	1	0	50	F_2: 0.50	F_7: 0.80
		2	50	100	0.47	0.75
		3	100	150	0.45	0.70
3	F_4	1	0	50	F_8: 1.70	F_9: 0.67
		2	50	110	1.80	0.70
		3	110	180	1.87	0.75
4	F_{13}	1	0	50	F_3: 1.18	F_{12}: 0.23
		2	50	90	1.15	0.25
		3	90	140	1.10	0.30
5	F_{14}	1	0	40	F_{11}: 0.37	F_{13}: 1.20
		2	40	80	0.35	1.25
		3	80	130	0.30	1.30
6	F_5	1	0	20	F_{14}: 1.15	
		2	20	45	1.10	
		3	45	75	1.02	

4.5 NUMERICAL RESULTS

In all of the examples, the initial values of the variables correspond to an incorrect subregion. We do that in order to test the ability of the algorithm for crossing boundaries until finding the correct subregion and iterating until obtaining the solution to the problem. Examples 1 through 3 were introduced by Zaher(1995). In that work, Zaher solves example 1 and 3 as optimization problems in order to test the performance of his approach to sequential quadratic programming. In this work we solve those problems as simulation problems by eliminating the objective function and defining fixed values for the degrees of freedom.

Table 4-4 Starting point and converged flowrates for Example 4-6.

Stream	Starting Point (lbmole/hr)	Converged Value (lbmole/hr)
F_1	47.50	47.50
F_2	21.25	19.85
F_3	69.00	57.75
F_4	25.00	23.35
F_5	50.00	36.52
F_6	37.50	34.94
F_7	34.00	31.76
F_8	52.50	39.70
F_9	16.75	15.65
F_{10}	1.700	1.58
F_{11}	16.80	14.06
F_{12}	15.00	12.55
F_{13}	60.00	50.22
F_{14}	48.00	40.17

The number of iterations that we used to obtained the solution of each of these examples is shown in Table 4-5. Some observations are:

- The fixed parameters and constants for examples 1 through 3 are the same as those given by Zaher (1995).
- For the fluid transition problem, the number of iterations reported corresponds to a diameter of 5 cm.
- For the example of the heat exchanger, the number of iterations reported is for an area

equal to 379.12 ft^2 and flowrate of cooling water equal to 1104.31 lbmole/hr.

- For the simple flash calculation, the number of iterations reported is for a temperature of 200 K (liquid phase region).

As expected, the combinatorial complexity of the example of the pipeline network (2.748779×10^{11} possible subregions) affects the effectiveness of the boundary crossing algorithm. The solution path performed 44 boundary analyses. Thus with 106 iterations, the average number of Newton steps per region entered is only of the order of two. For the rest of the examples, the algorithm performance is very encouraging. For the linear mass balance example, the number of possible subregions is 729, but the fact that the equations are linear facilitates the convergence of the algorithm. An advantage of the boundary crossing implementation is the problem size. Note the number of equations inside each subregion is always the minimum, and it is not affected by the combinatorial nature of the problem.

Table 4-5 Solving conditional models by using the boundary crossing algorithm.

Example	Reference	Number of Equations	Number of Disjunctions	Number of boundary analyses	Number of Iterations
Flow Transition (sonic-subsonic)	Zaher (1995)	5	1	1	10
Phase Equilibria	Zaher (1995)	12	3	5	17
Heat exchanger	Zaher (1995)	48	2	1	8
Pipeline network	Bullard and Biegler (1992)	98	38	44	106
Simple L-V flash	King (1980)	15	1	1	12
Linear mass balance	Grossmann and Turkay (1996)	13	6	2	8

4.6 CONCLUSIONS

Following a brief description of the boundary crossing algorithm, we have described the details of its practical implementation. The conditional modeling solver CMSlv is based on an server-client, object-oriented architecture. CMSlv executes a number of activities including:

- Configuration of the systems of variables and equations consistent with the restrictions imposed by the boundary expression.
- Creation of optimization subproblems at boundaries.
- Managing the logic of the algorithm
- Performing termination tests.
- Managing the interaction with external solvers; LRSlv for logical analysis, QRSlv for iterating within a subregion and CONOPT for the solution of the optimization subproblems at boundaries.

The CMSlv and LRSlv solvers have been attached to the ASCEND environment, whose modeling language provides the tools required for the representation of conditional models in an equation oriented manner. Finally, the modeling and solution of a number of examples show the scope of application of the algorithm and reveal its encouraging performance.

4.7 REFERENCES

Allan, B. A.; A More Reusable Modeling System; Ph.D. thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania,1997.

Bullard, L. G. and Biegler, L. T.; Iterative Linear Programming Strategies for Constrained Simulation. *Comput. Chem. Eng.*, 15(4):239–254, 1991.

Bullard, L. G. and Biegler, L. T.; Iterated Linear Programming Strategies for Nonsmooth Simulation: Continuous and Mixed-Integer Approach. *Comput. Chem. Eng.*, 16(10), 1992

Drud, A.; CONOPT: A GRG Code for Large Sparse Dynamic Nonlinear Optimization Problems. *Mathematical Programming*, 31, 153-191, 1985.

Grossmann, I. E. and Turkay, M.; Solution of Algebraic Systems of Disjunctive Equations. *Comput. Chem. Eng.*, 20:S339–44, 1996. Suppl. Part A.

King, C. J.; Separation Processes, Chemical Engineering Series, 2nd. Edition, McGrawHill, 77-79, 1980.

Raman, R. and Grossmann, I. E.; Symbolic Integration of Logic in Mixed-Integer Linear Programming Techniques for Process Synthesis. *Comput. Chem. Eng.*, 17(9):909–927, 1993.

Raman, R. and Grossmann, I. E.; Modeling and Computational Techniques for Logic Based Integer Programming. *Comput. Chem. Eng.*, 18(7):563–578, 1994.

Turkay, M. and Grossmann, I. E.; Logic-Based MINLP Algorithms for the Optimal Synthesis of Process Networks. Comput. Chem. Eng., 20(8):959–978, 1996.

Westerberg, A.W., Abbott, K. A., and Allan, B. A.; Plans for ASCEND IV: Our Next Generation Equational-Based Modeling Environment. Boston, Massachusetts, November 1994. AspenWorld'94.

Westerberg, A. W. and Director, S. W.; A Modified Least Square Algorithm for Solving Sparse n x n Sets of Nonlinear Equations. Technical report, Carnegie Mellon University, Engineering Design Research Center, EDRC-06-5-79, January 1979.

Westerberg, K.M.; Development of Software for Solving Systems of Nonlinear Equations. Technical Report. Carnegie Mellon University, Engineering Design Research Center, EDRC 05-36-89, 1989.

Zaher, J. J.; Conditional Modeling. Ph.D. thesis, Department of Chemical Engineering Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1995.

Zaher, J. J.; Conditional Modeling in an Equation-Based Modeling Environment. The Annual AIChE National Meeting, 1991. paper 138c.

Zaher, J. J.; Conditional Programming. The Annual AIChE National Meeting, March 1993.

CHAPTER 5 A COMPLEMENTARITY FORMULATION

In this chapter, we investigate the solving of conditional models using a complementarity formulation for representing algebraic systems of disjunctive equations. This formulation not only establishes the complementarity condition among equations belonging to different disjunctive terms but also enforces simultaneous satisfaction of all of the equations appearing within the same disjunctive term. This approach represents an alternative to MINLP formulations, avoiding discrete decisions; it also avoids the need for special procedural nonlinear techniques as required by the boundary crossing algorithm. We identify the advantages and disadvantages associated with the proposed formulation. The proposed complementarity representation performed reliably on several example problems where the number of equations in each disjunctive term is small.

5.1 COMPLEMENTARITY APPROACH

Over the last thirty years, the class of problems known as complementarity problems has become increasingly popular as a tool for addressing practical problems arising in mathematical programming, economics, engineering, and the sciences (Billups, 1995; Ferris and Pang, 1995). Several works have documented the basic theory, algorithms and applications of complementarity problems. Dirske and Ferris (1995b) give examples of how to formulate many popular problems as mixed complementarity problems (MCP). Billups (1995) describes the standard forms for the different classes of complementarity problems and proposes strategies which enhance the robustness of Newton-based methods for solving these problems. More (1994) formulates the complementarity problem as a nonlinear least square problem and gives convergence properties for his approach. In this chapter, we describe an extension to the standard complementarity formulation (Billups, 1995) for the representation of conditional models.

5.2 PROBLEM FORMULATION

In general, the nonlinear complementarity problem is expressed as the following set of equations and inequality constraints:

$$\begin{gathered} x_j \cdot r_j(\underline{x}) = 0 \\ x_j \geq 0 \qquad r_j(\underline{x}) \geq 0 \\ \forall j \in \{1 \dots n\} \end{gathered} \tag{5.1}$$

where n is the dimensionality of the vectors $\underline{x}$ and $\underline{r(x)}$. There is certain lack of symmetry in formulation (5.1). One of the functions is quite arbitrary while the other is the vector of variables. Many commonly occurring problems have a more general form:

$$\begin{gathered} r_{1_j}(\underline{x}) \cdot r_{2_j}(\underline{x}) = 0 \\ r_{1_j}(\underline{x}) \geq 0 \qquad r_{2_j}(\underline{x}) \geq 0 \\ \forall j \in \{1 \dots n\} \end{gathered} \tag{5.2}$$

which is called the vertical nonlinear complementarity problem (Ferris and Pang, 1995). It is possible, of course, to have more than two vectors of functions in the above equations:

$$\begin{array}{c} r_{1_j}(\underline{x}) \cdot r_{2_j}(\underline{x}) \ldots r_{k_j}(\underline{x}) = 0 \\ r_{i_j}(\underline{x}) \geq 0 \qquad \forall i \in [1 \ldots k], \forall j \in \{1 \ldots n\} \end{array} \tag{5.3}$$

For the case of a *single conditional equation* in a disjunctive statement, there exists an equivalent representation by using a standard complementarity formulation as follows:

$$\begin{bmatrix} r_i + g_i(\underline{x}) < 0 \\ r_i = 0 \end{bmatrix} \vee \begin{bmatrix} r_i + g_i(\underline{x}) \geq 0 \\ g_i(\underline{x}) = 0 \end{bmatrix} \Leftrightarrow \begin{array}{c} r_i \cdot g_i(\underline{x}) = 0 \\ r_i \geq 0 \qquad g_i(\underline{x}) \leq 0 \end{array} \tag{5.4}$$

A typical example of this equivalence can be found while representing the complementarity equations arising from the Karush-Kuhn-Tucker conditions of an optimization problem. There are also cases in which physicochemical transitions are complementary in nature and can be represented by such a formulation. For instance, the adiabatic compressible flow described by Zaher (1995) can be represented by an equivalent complementarity representation:

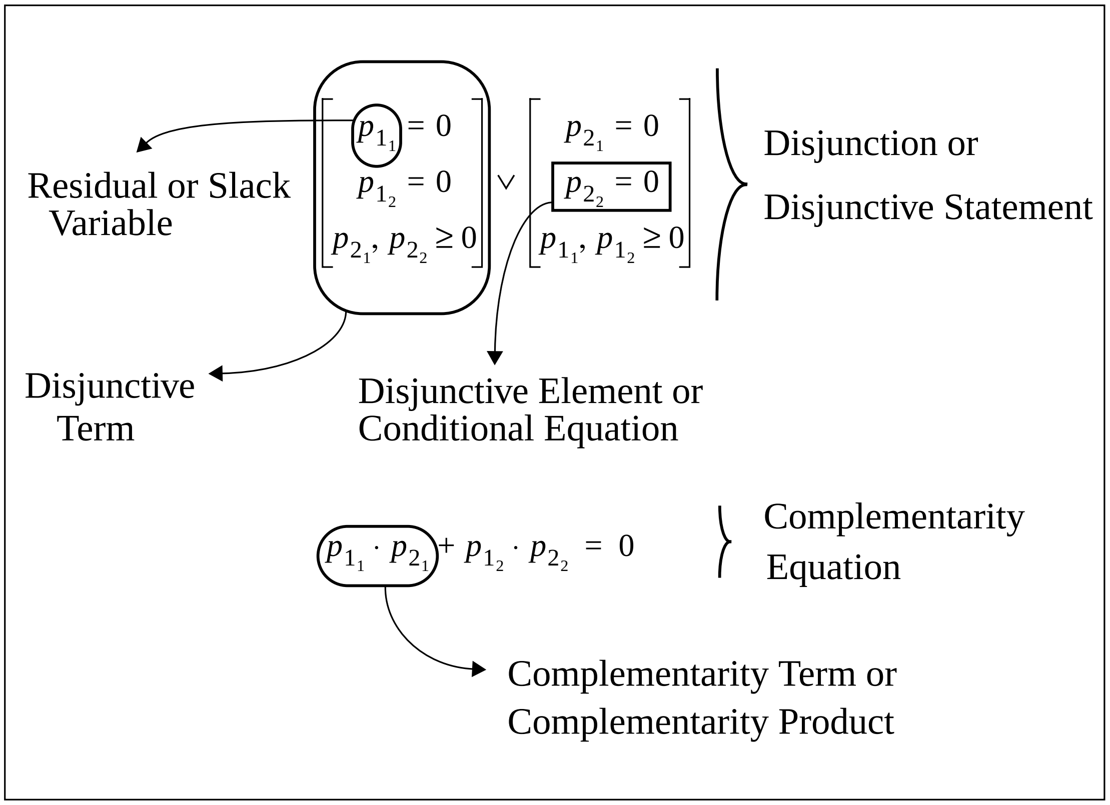
$$\begin{bmatrix} P_d - P_f < M_f - 1 \\ M_f - 1 = 0 \end{bmatrix} \vee \begin{bmatrix} P_d - P_f \geq M_f - 1 \\ P_d - P_f = 0 \end{bmatrix} \Leftrightarrow \begin{array}{c} (M_f - 1) \cdot (P_d - P_f) = 0 \\ M_f \leq 1 \qquad P_f \geq P_d \end{array} \tag{5.5}$$

On the other hand, if the disjunctive statement has *more than one equation in each disjunctive term*, as the example of the heat exchanger given also by Zaher (1995), the standard complementarity formulation is not equivalent to the disjunctive representation:

$$\begin{bmatrix} \sum\limits_{i \in C} x_{i_2} + \phi_1 < 1 \\ \phi_1 = 0 \\ \eta_2 = 0.5 \end{bmatrix} \vee \begin{bmatrix} \sum\limits_{i \in C} x_{i_2} + \phi_1 \geq 1 \\ \sum\limits_{i \in C} x_{i_2} = 1 \\ \eta_1 = 0.5 \end{bmatrix} \not\Leftrightarrow \begin{array}{c} \phi_1 \cdot \left(\sum\limits_{i \in C} x_{i_2} - 1 \right) = 0 \\ (\eta_2 - 0.5) \cdot (\eta_1 - 0.5) = 0 \\ \phi_1 \geq 0 \qquad \sum\limits_{i \in C} x_{i_2} \leq 1 \end{array} \tag{5.6}$$

Notice that in the disjunctive representation all the equations belonging to the same disjunctive term have to be satisfied simultaneously, a restriction which is not represented by the standard complementarity formulation. In the following section, we propose the representation of disjunctive sets of algebraic equations as a complementarity problem. The formulation described in this chapter not only establishes the complementarity condition among alternative sets of equations, but also enforces simultaneous satisfaction of all the equations in the solution set. It is important to mention that we are aware of many disadvantages associated with this representation, which we consider at the end of this section. Our motivation is that a complementarity formulation only requires the solution of one square system of nonlinear equations, avoiding all of the complications encountered in procedural techniques such as the boundary crossing algorithm and the discrete decision making of a MINLP solution. Before going further in the description of our approach, in Figure 5-1 we explain the terminology employed in this chapter.

Figure 5-1 Description of our terminology.


$\begin{bmatrix} p_{1_1} = 0 \\ p_{1_2} = 0 \\ p_{2_1}, p_{2_2} \ge 0 \end{bmatrix} \vee \begin{bmatrix} p_{2_1} = 0 \\ p_{2_2} = 0 \\ p_{1_1}, p_{1_2} \ge 0 \end{bmatrix}$
Residual or Slack Variable
Disjunction or Disjunctive Statement
Disjunctive Term
Disjunctive Element or Conditional Equation
$p_{1_1} \cdot p_{2_1} + p_{1_2} \cdot p_{2_2} = 0$
Complementarity Equation
Complementarity Term or Complementarity Product

5.2.1 COMPLEMENTARITY REPRESENTATION OF A CONDITIONAL MODEL

As in most of the complementarity approaches reported in the literature, in this work we assume the nondegeneracy of the solution to the complementarity problem. With this assumption we leave out all those cases in which the equations belonging to different terms of the same disjunctive statement are simultaneously satisfied. In addition, we will assume that the range of the conditional equations is positive; that is, we assume that we can rearrange the system of equations so that the residuals of the conditional equations belonging to a non-solution disjunctive term will always be positive.

The formulation presented here is an extension of that presented by More(1994), who reformulates the nonlinear complementarity problem (5.1) as:

$$\begin{gathered} \underline{r}(\underline{x}) = \underline{p} \\ P \cdot \underline{x} = \underline{0} \\ \underline{x} \geq 0 \qquad \underline{p} \geq 0 \end{gathered} \tag{5.7}$$

where $\underline{p}$ is a vector of residual variables and P is the diagonal matrix diag(p_i).

For the purpose of illustration, consider the example of the heat exchanger given by (5.6). Defining positive residuals p_{i_j} for each of the conditional equations:

$$\begin{aligned} \phi_1 &= p_{1_1} & \qquad 1 - \sum_{i \in C} x_{i_2} &= p_{2_1} \\ \eta_2 - 0.5 &= p_{1_2} & \qquad 0.5 - \eta_1 &= p_{2_2} \end{aligned} \tag{5.8}$$

we then represent the disjunctive statement in terms of these residuals:

$$\begin{bmatrix} p_{1_1} = 0 \\ p_{1_2} = 0 \\ p_{2_1}, p_{2_2} \geq 0 \end{bmatrix} \vee \begin{bmatrix} p_{2_1} = 0 \\ p_{2_2} = 0 \\ p_{1_1}, p_{1_2} \geq 0 \end{bmatrix} \tag{5.9}$$

Notice that we have arranged the conditional equations in (5.8) in such a way that all the variables p_{i_j} are always positive. In order to do that, we used the physical insight given by the nature of the problem. In terms of the residual variables p_{i_j}, the standard complementarity formulation is given by:

$$\begin{aligned} p_{1_1} \cdot p_{2_1} &= 0 \\ p_{1_2} \cdot p_{2_2} &= 0 \\ p_{i_j} \geq 0 \qquad i \in [1 \ldots 2] & \qquad j \in [1 \ldots 2] \end{aligned} \tag{5.10}$$

The disjunctive statement in (5.9) requires either both p_{1_1} and p_{1_2} or both p_{2_1} and p_{2_2} simultaneously to be zero. That is not a restriction included in the standard complementarity formulation (5.10). We propose the following formulation to represent the disjunctive statement instead:

$$\begin{aligned} p_{1_1} \cdot p_{2_1} + p_{1_2} \cdot p_{2_2} &= 0 \\ p_{1_1} \cdot p_{2_2} + p_{1_2} \cdot p_{2_1} &= 0 \\ p_{i_j} \geq 0 \qquad i \in [1 \ldots 2] & \qquad j \in [1 \ldots 2] \end{aligned} \tag{5.11}$$

Since the variables p_{i_j} are all positive, the set of equations in (5.11) not only contains the complementarity condition given by the standard representation (5.10) but also enforces the simultaneous satisfaction of all the equations defined in the same terms of the disjunction. Also, it is important to realize that the inequality constraints in (5.11) are only bounds in the residual variables, and we can use them to guide our search for a solution to the resulting square system of equations:

- The original disjunctive statement in (5.6) provides 2 equations in either of its cases.
- The complementarity formulation in (5.8) and (5.11) provides 6 equations but introduces 4 new variables, also a net of 2 equations.

For cases in which the domain of validity is given by inequality constraints, we reformulate the problem by adding two slacks to each inequality and express the complementarity condition between these slack variables. Consider for example the laminar-turbulent flow transition given by:

$$\begin{bmatrix} Re = 64/f \\ Re \leq 2100 \end{bmatrix} \vee \begin{bmatrix} Re = (0.206307/f)^4 \\ Re > 2100 \end{bmatrix} \quad \textbf{(5.12)}$$

If we define residuals for the equalities and slack variables for the inequalities:

$$Re - 64/f = p_{1_1} \qquad Re - (0.206307/f)^4 = p_{2_1}$$
$$Re = 2100 + p_{1_2} - p_{2_2} \quad \textbf{(5.13)}$$

then the disjunctive statement in terms of residuals and slacks and, therefore, the complementarity equations, are exactly the same as those given by (5.11) for the previous example. Notice that the inequalities become an active part of the system of equations. Also, one of the complementarity equations contains a complementarity product between the two slacks in the inequality, a requirement to avoid multiple solutions.

The result of applying our formulation is a square system of nonlinear equations (including complementarity equations) subject to the positiveness of the variables p_{i_j}.

The proposed complementarity representation has the following properties:

1. The number of complementarity equations is equal to the number of equations in each term of the disjunction to maintain the same number of degrees of freedom as in the original problem.
2. Each residual variable is multiplied by every other residual variable in all of the other terms of the disjunction. Thus, we will ensure the simultaneous satisfaction of all the equations in at least one of the disjunctive terms and avoid spurious solutions to the problem.
3. In the example, there are several ways in which we could have accommodated the four complementarity terms in the two complementarity equations. The way in which we have distributed the bilinear terms over the complementarity equations is intended to decrease the possibility of having numerical singularities in the Jacobian of the system while using an iterative solver based on Newton and quasi-Newton methods. We must

avoid having two residual variables from the same disjunctive term being multiplied in two complementarity equations by the same set of residual variables from another disjunctive term. Examine Figure 5-2. We show a set of poorly formulated complementarity equations and the rows of the Jacobian corresponding to those equations. Note that the equations in Figure 5-2 contain the same four complementarity terms as the formulation given by (5.11), but they are grouped differently. In the case presented, if the solution to the problem is $p_{2_1} = p_{2_2} = 0$, rows 1 and 2 of the Jacobian become numerically dependent as the Newton method approaches the solution. If $p_{2_1} + p_{2_2} \sim 0$, row 2 is a multiple of row 1 by the factor p_{1_2} / p_{1_1}. This situation does not occur for the complementarity equations we propose in (5.11).

Figure 5-2 Numerical singularities in complementarity equations.

$$\begin{array}{l} p_{1_1} \cdot p_{2_1} + p_{1_1} \cdot p_{2_2} = 0 \\ p_{1_2} \cdot p_{2_1} + p_{1_2} \cdot p_{2_2} = 0 \end{array} \Rightarrow \begin{array}{l} p_{1_1} \cdot (p_{2_1} + p_{2_2}) = 0 \\ p_{1_2} \cdot (p_{2_1} + p_{2_2}) = 0 \end{array}$$

	p_{1_1}	p_{2_1}	p_{1_2}	p_{2_2}
1	$p_{2_1} + p_{2_2}$	p_{1_1}	0	p_{1_1}
2	0	p_{1_2}	$p_{2_1} + p_{2_2}$	p_{1_2}

Next, we formally describe how to obtain a complementarity formulation including all of the properties outlined above. We first consider the case in which the disjunctive statement contains two terms and then we extend the analysis to any number of disjunctive terms.

5.2.1.1 Representing the Disjunctive Statements in terms of Positive Residual or Slack Variables

Before generating the set of complementarity equations, it is necessary to define positive residual variables (or slack variables for inequalities) for the conditional equations and to represent the disjunctive statements in terms of these positive variables. This task has to be accomplished disregarding the number of terms in each disjunctive statement. As a matter of fact, one of our assumptions here is that the system of equations can be rearranged to obtain this representation. For instance, for the simplest case of one disjunction with two terms (the index k is omitted for simplicity), given the disjunctive set of algebraic equations:

$$\begin{gathered} \underline{h}(\underline{x}) = 0 \\ \begin{bmatrix} r_{1_j}(\underline{x}) = 0 \\ g_l(\underline{x}) \le 0 \end{bmatrix} \vee \begin{bmatrix} r_{2_j}(\underline{x}) = 0 \\ g_l(\underline{x}) \ge 0 \end{bmatrix} \qquad \begin{matrix} \forall j \in [1\ldots\beta] \\ \forall l \in [1\ldots\gamma] \end{matrix} \end{gathered} \tag{5.14}$$

we reformulate the problem as:

$$\begin{gathered} \underline{h}(\underline{x}) = 0 \\ r_{1_j}(\underline{x}) - p_{1_j} = 0 \\ r_{2_j}(\underline{x}) - p_{2_j} = 0 \qquad \forall j \in [1\ldots\beta], l \in [\beta+1\ldots\beta+\gamma] \\ g_{l-\beta}(\underline{x}) - p_{1_l} + p_{2_l} = 0 \\ \begin{bmatrix} p_{1_q} = 0 \\ p_{2_q} \ge 0 \end{bmatrix} \vee \begin{bmatrix} p_{2_q} = 0 \\ p_{1_q} \ge 0 \end{bmatrix} \qquad \forall q \in [1\ldots\beta+\gamma] \end{gathered} \tag{5.15}$$

5.2.1.2 2-Term Disjunctive Statements

The set of complementarity equations in terms of the positive variables p_{i_q} equivalent to the disjunctive statement given in (5.15) is:

$$\sum_{t=1}^{\beta+\gamma-s} p_{1_t} \cdot p_{2_{t+s}} + \sum_{t=\beta+\gamma-s+1}^{\beta+\gamma} p_{1_t} \cdot p_{2_{t+s-\beta-\gamma}} = 0 \qquad \forall s \in [0 \ldots \beta+\gamma-1]$$
$$p_{i_q} \geq 0 \qquad \forall i \in [1 \ldots 2], q \in [1 \ldots \beta+\gamma] \tag{5.16}$$

Hence, the resulting nonlinear system of equations is:

$$h(\underline{x}) = 0$$
$$\begin{pmatrix} r_{1_j}(\underline{x}) - p_{1_j} \\ r_{2_j}(\underline{x}) - p_{2_j} \\ g_{l-\beta}(\underline{x}) - p_{1_l} + p_{2_l} \end{pmatrix} = 0 \qquad \forall j \in [1 \ldots \beta], l \in [\beta+1 \ldots \beta+\gamma]$$
$$\sum_{t=1}^{\beta+\gamma-s} p_{1_t} \cdot p_{2_{t+s}} + \sum_{t=\beta+\gamma-s+1}^{\beta+\gamma} p_{1_t} \cdot p_{2_{t+s-\beta-\gamma}} = 0 \qquad \forall s \in [0 \ldots \beta+\gamma-1]$$
$$p_{i_q} \geq 0 \qquad \forall i \in [1 \ldots 2], q \in [1 \ldots \beta+\gamma] \tag{5.17}$$

Note that the resulting nonlinear system of equations is square, and is subject to the bounds in the residual and slack variables.

The generation of the complementarity equations given by (5.16) is illustrated in Figure 5-3. Basically, in a complementarity equation each residual variable of one disjunctive term is multiplied by one residual variable of the other disjunctive term, and, for successive complementarity equations, the order of the residual variables in the second term is successively shifted by one.

Every possible complementarity term resulting from the multiplication of positive residual variables belonging to different disjunctive terms is included in complementarity equations (5.16). This feature ensures that, in order to satisfy the complementarity equations, all the residual variables of at least one disjunctive term have to be simultaneously zero. The proof that (5.16) satisfies at least one disjunctive term is as follows. Assume that, in each of the disjunctive terms, there is only one nonzero residual variable. Since all of the possible complementarity terms exist in the complementarity equations, a complementarity term containing those residual variables must exist. Find the complementarity term that contains just these nonzero residual variables. Since the product of those variables will be nonnegative, that term will force the complementarity equation in which it exists to be greater than zero, *i.e.*, it will not be satisfied. To be satisfied, at least one of the residual variables in the complementarity term must be zero, contradicting our original assumption. Thus, in order to satisfy the complementarity equations, at least one disjunctive term must have all its residual variables equal to zero. In other words, a complete set of conditional equations in at least one of the disjunctive terms will be satisfied.

Figure 5-3 Generation of complementarity equations in a two-term disjunction.

$$\begin{bmatrix} p_{1_1} = 0 \\ p_{1_2} = 0 \\ p_{1_3} = 0 \\ \underline{p}_2 \ge 0 \end{bmatrix} \vee \begin{bmatrix} p_{2_1} = 0 \\ p_{2_2} = 0 \\ p_{2_3} = 0 \\ \underline{p}_1 \ge 0 \end{bmatrix}$$

$$\beta + \gamma = 3 \qquad \Leftrightarrow$$

1) $s = 0$

$$\begin{tikzcd} p_{1_1} \arrow[r] & p_{2_1} \\ p_{1_2} \arrow[r] & p_{2_2} \\ p_{1_3} \arrow[r] & p_{2_3} \end{tikzcd}$$

$$p_{1_1} \cdot p_{2_1} + p_{1_2} \cdot p_{2_2} + p_{1_3} \cdot p_{2_3} = 0$$

2) $s = 1$

$$\begin{tikzcd} p_{1_1} \arrow[dr] & p_{2_1} \\ p_{1_2} \arrow[dr] & p_{2_2} \\ p_{1_3} \arrow[ddr] & p_{2_3} \\ - - - & - - - \\ p_{1_1} & p_{2_1} \end{tikzcd}$$

3) $s = 2$

$$\begin{tikzcd} p_{1_1} \arrow[ddr] & p_{2_1} \\ p_{1_2} \arrow[dddr] & p_{2_2} \\ p_{1_3} \arrow[dddr] & p_{2_3} \\ - - - & - - - \\ p_{1_1} & p_{2_1} \\ p_{1_2} & p_{2_2} \end{tikzcd}$$

$$p_{1_1} \cdot p_{2_2} + p_{1_2} \cdot p_{2_3} + p_{1_3} \cdot p_{2_1} = 0 \qquad p_{1_1} \cdot p_{2_3} + p_{1_2} \cdot p_{2_1} + p_{1_3} \cdot p_{2_2} = 0$$

$$p_{i_j} \ge 0$$

$$\forall i \in [1, 2], j \in [1 \ldots 3]$$

As a consequence of the previous analysis, if the complete set of nonlinear equations (including the complementarity equations) is satisfied, then the solution vector $\hat{x}$ will correspond to a consistent solution to the conditional model. Moreover, if we assume uniqueness of the solution to the conditional model, then the vector $\hat{x}$ will be such a unique solution.

Another property of the complementarity set of equations given by (5.16) is that, in every complementarity equation, all the residual variables appear, and each of them appears only once. This is intended to decrease the possibility of having numerical singularities in the Jacobian of the system, as explained before. In fact, by analyzing the Jacobian of the formulation (5.16) under the assumption of nondegeneracy of the solution, it can be shown that the possibility of having numerical singularities is eliminated. Hence, if the solution is not on a boundary, the residuals in the equations of the disjunctive set not corresponding to the solution are expected to be different from zero, and, therefore, they will provide a pivot in the Jacobian matrix for all the complementarity equations (note that the number of positive residuals is equal to the number of complementarity equations).

5.2.1.3 Generalization to any number of terms in the disjunctive statement

When the disjunctions contain more than two terms, we generate the complementarity equations by applying recursively the same equation given in (5.16). We assume again that we can obtain a disjunctive statement in terms of positive residual variables. Consider the following simple example of a disjunction with three terms:

$$\begin{bmatrix} p_{1_1} = 0 \\ p_{1_2} = 0 \\ \underline{p}_2, \underline{p}_3 \geq 0 \end{bmatrix} \vee \begin{bmatrix} p_{2_1} = 0 \\ p_{2_2} = 0 \\ \underline{p}_1, \underline{p}_3 \geq 0 \end{bmatrix} \vee \begin{bmatrix} p_{3_1} = 0 \\ p_{3_2} = 0 \\ \underline{p}_1, \underline{p}_2 \geq 0 \end{bmatrix} \qquad \textbf{(5.18)}$$

We apply equation (5.16) to the first two disjunctive terms:

$$\begin{bmatrix} p_{1_1} = 0 \\ p_{1_2} = 0 \end{bmatrix} \vee \begin{bmatrix} p_{2_1} = 0 \\ p_{2_2} = 0 \end{bmatrix} \vee \begin{bmatrix} p_{3_1} = 0 \\ p_{3_2} = 0 \end{bmatrix} \Leftrightarrow \begin{bmatrix} p_{1_1} \cdot p_{2_1} + p_{1_2} \cdot p_{2_2} = 0 \\ p_{1_1} \cdot p_{2_2} + p_{1_2} \cdot p_{2_1} = 0 \end{bmatrix} \vee \begin{bmatrix} p_{3_1} = 0 \\ p_{3_2} = 0 \end{bmatrix} \tag{5.19}$$
$$p_{i_j} \geq 0 \qquad \forall i \in [1\ldots3],\ j \in [1\ldots2]$$

and do it again for the resulting two-term disjunction:

$$\begin{aligned} (p_{1_1} \cdot p_{2_1} + p_{1_2} \cdot p_{2_2}) \cdot p_{3_1} + (p_{1_1} \cdot p_{2_2} + p_{1_2} \cdot p_{2_1}) \cdot p_{3_2} &= 0 \\ (p_{1_1} \cdot p_{2_1} + p_{1_2} \cdot p_{2_2}) \cdot p_{3_2} + (p_{1_1} \cdot p_{2_2} + p_{1_2} \cdot p_{2_1}) \cdot p_{3_1} &= 0 \\ p_{i_j} \geq 0 \qquad \forall i \in [1\ldots3],\ j \in [1\ldots2] & \end{aligned} \tag{5.20}$$

A complementarity set of equations obtained in this form still will include the properties outlined before for a two-term disjunction:

1. It will result in a square system of equations.
2. In order to satisfy the complementarity equations, all the equations of at least one set of conditional equations have to be simultaneously satisfied.
3. Under the assumption of nondegeneracy, the Jacobian of the system of equations can be shown to be nonsingular.

The reasoning employed to prove the previous statements is the same as that in the case of a two-term disjunction discussed above.

5.2.1.4 About the Complementarity Formulation

The complementarity formulation is not without problems:

1. First of all, the problem grows quickly. If (β+γ) is the number of equations in each term of the disjunction and D is the number of terms in the disjunction, the number of equations representing the disjunctive statement in the complementarity formulation is (β+γ)(D+1). That number includes both the complementarity equations and the equations defining the positive residual variables.
2. Numerical singularities still arise for cases in which the solution resides on a boundary. That is the main reason for the assumption of nondegeneracy.
3. The reformulation of a conditional model as a complementarity problem is restricted

to cases in which the range of the conditional equations is positive. We consider that as the major restriction of this approach. In principle, a cure for this limitation is to define complementarity variables p_{i_q} which can take negative and positive values, and then to use the square of these complementarity variables in the complementarity equations. However, it is known that the use of square terms in complementarity equations is not convenient since the possibility of introducing numerical singularities is even greater. Recall that the derivative of the square of a variable with value zero is also zero.

4. The number of bilinear terms (or terms including products among variables) incorporated in each equation also grows with the number of equations in each term of the disjunction. The combinatorial nature of the problem is encapsulated here.
5. The performance of optimization techniques is badly affected by the introduction of nonconvexities (multiplication among variables) to the system of equations.

In general, we are presenting this approach as a favorable alternative when the number of equations in each disjunctive term is small.

5.3 EXAMPLES

We used the examples of algebraic systems of disjunctive equations described in Chapter 4 for testing the proposed complementarity formulation. Appendix D presents the complementarity equations (or a representative part of them) for each of those examples.

In examples 4.1 through 4.4 the disjunctive statements contain only two terms, and we generated the complementarity equations by strictly using the formulation proposed in (5.16). The degree of complexity increases in examples 4.5 and 4.6 since the disjunctive statements contain three terms. In those cases, we added two residual variables to the conditional equations in one of the disjunctive terms. Moreover, in examples 4.5 and 4.6 we do not apply the formulation proposed for three-term disjunctions. Instead, we use those examples to show how sometimes the specific structure of the disjunctive statement can be used to simplify the resulting system of complementarity equations.

In all the examples the number of equations in each disjunctive term is very small, and, as a consequence, the complexity of the equations involving products among variables is not as bad as can be expected with problems of larger size.

As in Chapter 4, in this chapter we also solved examples 1 and 3 as simulation problems. The initial values for the variables are the same as those given, if reported, by the reference.

5.4 SOLVING THE COMPLEMENTARITY FORMULATION OF A CONDITIONAL MODEL

There exists a whole body of literature for the solution of nonlinear complementarity problems. In one of the most recent approaches, Dirske and Ferris (1995a) developed a non-monotone stabilization scheme for mixed complementarity problems (MCP) and implemented such an approach in the PATH solver. Some other approaches to the solution of MCPs involve the use of quadratic programming based techniques (Billups and Ferris, 1996) and homotopy based algorithms (Billups, 1998). As stated, a common feature of the previous approaches is that they require the complementarity problem to be reformulated as a MCP. However, a systematic reformulation of our complementarity problem (5.17) into a MCP is yet to be discovered. Such a systematic reformulation is desired because it would make suitable the application of a large number of existing codes and numerical techniques such as the ones described above. Future work should be conducted in this direction.

In this section, we investigate two approaches to the solution of our complementarity formulation representing a conditional model. Since the result of applying the complementarity formulation is one square nonlinear system of equations, the first approach is just to use a conventional nonlinear solver. The second approach consists in solving the complementarity problem based on pivotal techniques similar to those proposed by Lemke (1965).

Later, in Chapter 6, we investigate the solution of the complementarity formulation by using interior point methods.

5.4.1 SOLVING BY USING A CONVENTIONAL SOLVER

To solve the examples described in section 5.3, we used the ASCEND solver QRSlv which applies the modified Levenberg-Marquardt algorithm given by Westerberg and Director (1979).

5.4.1.1 Numerical Results

The number of iterations that we used to obtain the solution of each of these examples is shown in Table 5-1. Some observations are:

- The fixed parameters and constants for Examples 1 through 3 are the same as those giving by Zaher (1995).
- For the fluid transition problem, we ran simulations for values of the diameter of the pipe between 2 cm and 9 cm, such that we could make sure that both of the alternative cases are reached by using the complementarity representation. The number of iterations reported corresponds to the diameter of 5 cm.
- For the example of the heat exchanger, we ran simulations for values of area between 250 ft^2 and 1104 ft^2 and values of flowrates between 250 lbmole/hr and 380 lbmole/hr, ranges analyzed by Zaher while finding an optimal solution. The number of iterations reported is for an area equal to 379.12 ft^2 and flowrate of cooling water equal to 1104.31 lbmole/hr. There is no special reason for the selection of those values.
- For the simple flash calculation, the number of iterations reported is for the value of temperature of 200 K (liquid phase region). The number of iterations was roughly of the same order for other values of temperature spread over the subcooled through the superheated range.

Since the hardware and the technique that we are using to get the solution is different, we are comparing neither time nor number of iterations with other works. Still we consider it important to make remarks about some differences of the alternative approaches for solving conditional models.

Table 5-1 Solving the complementarity problems by using a conventional solver.

Example	Reference	Number of Equations	Number of Disjunctions	Number of Complementarity Equations	Iterations
Flow Transition (sonic-subsonic)	Zaher (1995)	7	1	1	8
Phase Equilibria	Zaher (1995)	18	3	3	10
Heat exchanger	Zaher (1995)	56	2	4	11
Pipeline network	Bullard and Biegler (1992)	250	38	76	24
Simple L-V flash	King (1980)	23	1	4	25
Linear mass balance	Grossmann and Turkay (1996)	81	6	34	25

In the example of the pipeline network, the result of the complementarity representation is one nonlinear system containing 250 equations, 76 of them containing 2 bilinear terms. In the same example, the number of boundaries in the boundary crossing algorithm is 38 (that means 2^{38}=2.7 x 10^{11} possible subregions) and the nonlinear system to be solved in every subregion would contain 98 equations. The combinatorial complications present in examples like this clearly represent a disadvantage for the boundary crossing algorithm. On the other hand, the reverse situation is also possible, and the boundary crossing may be clearly a better option than the complementarity formulation. For example, in a problem with only one disjunction, but 20 equations in each of the terms of the disjunction, we may not be able to solve a nonlinear system in which 20 of the equations contain 20 bilinear terms each. However, the number of subregions in the boundary crossing algorithm would be only 2, and the possibility of applying that algorithm efficiently would be much greater.

In the example of the linear mass balance, Turkay and Grossmann (1996) solve the problem by using a mixed-integer approach. The resulting MILP contains 18 binary variables, 66 continuous variables and 89 linear equations. In the complementarity formulation, the nonlinear system contains 81 equations, 47 are linear, but the remaining

34 contain complementarity products. The size of the problem is very similar, the difference will be in either using a branch and bound search in the MILP or dealing with the complementarity equations in the solution of one square nonlinear system of equations.

5.4.1.2 About the Solution with a Conventional Nonlinear Solver

The Levenberg-Marquardt technique performed well in the set of examples used in this work. This technique is preferred because it is a least squares method which may help to overcome numerical singularities arising from the complementarity formulation. In general, however, it is known that problems involving complementarity equations affect the performance of conventional nonlinear techniques. In particular, Newton steps cannot be calculated if the Jacobian matrix of the system of equations is singular. Also, numerical difficulties associated with bad scaling are commonly encountered while solving complementarity problems. This lack of robustness of the use of a conventional nonlinear solver in the solution of our complementarity formulation motivated the use of interior point methods. We describe such work in Chapter 6.

5.4.2 SOLVING BY USING PIVOTAL TECHNIQUES

In the late 1960's, Lemke (1965) and Cottle and Dantzig (1968) developed the complementarity pivot theory for finding the solution of linear complementarity problems. Given a real k-vector q and a $k \times k$ real matrix M, the solution to the linear complementarity problem is given by the k-vectors w and z which satisfy:

$$\begin{gathered} w = q + M \cdot z \\ w \cdot z = 0 \\ w \geq 0 \qquad z \geq 0 \end{gathered} \tag{5.21}$$

5.4.2.1 Complementarity Pivot Theory for Linear Complementarity Problems

Consider the system of linear equations:

$$\begin{gathered} w = q + M \cdot z \\ w \geq 0 \qquad z \geq 0 \end{gathered} \tag{5.22}$$

For *i=1..k* the corresponding variables z_i and w_i are called complementary, and each is the complement of the other. A complementary solution of (5.22) is a pair of vectors satisfying (5.22) and

$$w_i \cdot z_i = 0 \qquad i = 1 \ldots k \tag{5.23}$$

Following the linear programming methodology, the independent variables of (5.22) are called *nonbasic*, while the dependent variables are called *basic*. The basic variables are said to constitute a *basis*. Also, all the nonbasic variables are set to zero.

A complementary basic feasible solution of (5.22) is one in which the complement of each basic variable is nonbasic. The goal is to obtain a basic feasible solution with this property.

In Lemke's method (1965), an extra column (called covering vector) is added to the matrix *M* along with an artificial variable λ. Typically, the covering vector is taken to be the vector of all ones, *e*. Thus, (5.22) is replaced by:

$$\begin{gathered} w = q + e \cdot \lambda + M \cdot z \\ w \geq 0 \qquad z \geq 0 \qquad \lambda \geq 0 \end{gathered} \tag{5.24}$$

In the start of the algorithm, the variables *z* are nonbasic while λ is set to

$$min\{\lambda | \lambda \geq 0, e \cdot \lambda + q \geq 0\} \tag{5.25}$$

Note that λ and the covering vector are introduced to achieve feasibility for the augmented system (5.24) while also maintaining complementarity in the original variables. Hence, the selection of λ in (5.25) leads to an initial complementary basic feasible solution of (5.24). However, also note that the variables *z* and *w* are a solution of (5.21) only if $\lambda=0$. In general, λ will be basic in the initial basic feasible solution with the value $\lambda_0>0$ obtained from (5.25). Thus, Lemke's method specifies pivoting rules which determine a sequence of variables entering and leaving the basis and a sequence of basic feasible solutions which maintain the complementarity of *z* and *w*. Essentially, Lemke's pivoting rule states that a variable reaching its lower bound will leave the basis, and that the complement of the variable leaving the basis will be the next variable entering the basis. The algorithm terminates successfully when a pivot results in λ leaving the basis at value zero.

5.4.2.2 Solving the Complementarity Representation of Conditional Models by Using Lemke's Pivoting Rules

Some researchers have investigated the solution of nonlinear complementarity problems by using techniques based on Lemke's pivoting rules. Dirske and Ferris (1995a) reformulate the mixed complementarity problem in such a way that a first order approximation of it results in a linear complementarity problem. Hence, they use a Newton-based technique in which a linear complementarity problem is solved at each iteration. They also provide a proof of convergence for their approach.

For the case of our complementarity representation, a theoretically sound extension of Lemke's pivotal technique has not yet been discovered. None the less, we studied the solution of the complementarity representation of conditional models with a heuristic approach based on Lemke's pivoting rules. Such an approach is as follows. Consider the representation of the disjunctive set of equations given in terms of the residual and slack variables given by (5.15):

$$\underline{h}(\underline{x}) = 0$$

$$\begin{aligned} r_{1_j}(\underline{x}) - p_{1_j} &= 0 \\ r_{2_j}(\underline{x}) - p_{2_j} &= 0 \qquad \forall j \in [1\ldots\beta],\ l \in [\beta+1\ldots\beta+\gamma] \\ g_{l-\beta}(\underline{x}) - p_{1_l} + p_{2_l} &= 0 \end{aligned}$$

$$\begin{bmatrix} p_{1_q} = 0 \\ p_{2_q} \geq 0 \end{bmatrix} \vee \begin{bmatrix} p_{2_q} = 0 \\ p_{1_q} \geq 0 \end{bmatrix} \qquad \forall q \in [1\ldots\beta+\gamma]$$

1. Here, the complementary variables are the variables p_{i_q}. Also, the variables p_{1_q} in (5.15) are the complement of the variables p_{2_q}.
2. We start by defining the complementary variables in one of the disjunctive terms of (5.15) as basic and the complementary variables in the other disjunctive terms as nonbasic. Since the nonbasic variables are set to zero, the complementarity conditions given by the complementarity equations in (5.16) are thus satisfied.

3. We then look for a solution to the nonlinear problem by using a conventional nonlinear solver. If during the iterative process a basic complementary variable reaches its lower bound, then a complementarity pivot analogous to a Lemke's is taken. In our case, all the complementary variables in the same disjunctive term as the variable reaching its bounds are simultaneously removed from the basis. The complement of such variables (the variables in the other disjunctive term of the disjunction) become basic and the iterative process continues. Such a complementarity pivot is taken only if a step in the new basis results in the decrease of the norm of the residuals of the overall system of equations with respect to the point prior to the complementarity pivot. Otherwise, we only project the basic variable reaching its bound to such a bound and continue the iterative process without the change of basis.
4. When the number of disjunctive terms in the disjunction is greater than two, we need to decide which alternative set of variables are going to constitute the new basis after a complementarity pivot. This is consistently accomplished by defining boundary expressions (similar to those in the boundary crossing algorithm) in terms of the complementary variables. The boundary expressions are not a part of the system of equations to be solved, but only a means to find the next set of basic variables without ambiguity.

5.4.2.3 Numerical Results

The approach described in the previous section was also applied to solve the complementarity formulation of the examples described in Chapter 4. The number of iterations that we used to obtain the solution of each of these examples is shown in Table 5-2:

5.4.2.4 Discussion

It should be noticed that we give no theoretical guarantee of convergence for the solution of the complementarity problems by using Lemke's pivoting rules. Even so, the results obtained from the numerical experiments show a surprising effectiveness.

Table 5-2 Solving the complementarity problems by using pivotal techniques.

Example	Reference	Number of Complementarity Pivots Taken	Iterations
Flow Transition (sonic-subsonic)	Zaher (1995)	1	7
Phase Equilibria	Zaher (1995)	1	7
Heat exchanger	Zaher (1995)	1	8
Pipeline network	Bullard and Biegler (1992)	12	27
Simple L-V flash	King (1980)	1	11
Linear mass balance	Grossmann and Turkay (1996)	2	9

Next, we show a comparison between the boundary crossing algorithm and the pivotal technique described here:

1. In the boundary crossing algorithm, only the equations corresponding to the current alternative of each disjunction are considered at each iteration. In the extended Lemke technique we simultaneously consider all of the alternatives, but we relax the equations of all but one alternative of each disjunction. Thus, in the extended Lemke technique we are also enforcing the satisfaction of only one alternative of each disjunction at each iteration.
2. The crossing of a boundary in the boundary crossing algorithm involves a change in the system of equations constituting the problem. In the extended Lemke technique, a complementarity pivot causes an equivalent effect.
3. In boundary crossing, the solution of the optimization subproblem at the boundary provides a solid basis to decide whether a boundary should be crossed or not and ensures that cycling is avoided during the iteration process. In the extended Lemke technique, a proof against cycling in our complementarity formulation is not yet available.

Accordingly, we assert that the boundary crossing provides a generalization of Lemke's pivotal technique for the solution of our complementarity problem; generalization in which a mathematical criterion is given to decide whether a complementarity pivot should be taken or not.

The results obtained by using our heuristic extended Lemke technique, Table 5-2, make us believe that it may be not necessary to find an optimal descent direction in a boundary analysis of the boundary crossing algorithm described in Chapter 4. In other words, perhaps any step which reduces the residuals of the systems of equations in the neighborhood of the boundary (any descent as opposed to optimal descent) can give a reasonable movement at the boundary.

5.5 SUMMARY

We have proposed and tested a new representation of conditional models as complementarity problems. In order to obtain the complementarity formulation, we rely on the assumption that the sets of conditional equations can be rearranged so that the disjunctive statements can be represented in terms of positive residual and slack variables. We show that the formulation described in this paper does not introduce spurious solutions to the problem, and that under the assumption of nondegeneracy, it will not introduce numerical singularities to the Jacobian matrix. We also mentioned some of the weaknesses and advantages of this approach. We solved the complementarity problems described in Chapter 4 and Appendix D by using a conventional nonlinear solver and Lemke's pivoting rules. The number of iterations employed for all of the examples solved here makes the complementarity formulation appear as an interesting tool. Finally, we described how the boundary crossing algorithm can be considered as a generalization of the extended Lemke technique used here.

5.6 REFERENCES

Billups, S. C.; Algorithms for Complementarity Problems and Generalized Equations. Ph.D. thesis, University of Wisconsin-Madison, August 1995.

Billups, S. C.; A Homotopy Based Algorithm for Mixed Complementarity Problems. Technical Report, Department of Mathematics, University of Colorado, 1998.

Billups, S. C. and Ferris, M. C.; QPCOMP: A Quadratic Programming Based Solver for Mixed Complementarity Problems. Technical Report, Department of Mathematics, University of Colorado, 1996.

Bullard, L. G. and Biegler, L. T.; Iterative Linear Programming Strategies for Constrained Simulation. *Comput. Chem. Eng.*, 15(4):239–254, 1991.

Bullard, L. G. and Biegler, L. T.; Iterated Linear Programming Strategies for Nonsmooth Simulation: Continuous and Mixed-Integer Approach. *Comput. Chem. Eng.*, 16(10), 1992.

Cottle, R. W. and Dantzig, G. B.; Complementarity Pivot Theory of Mathematical Programming. *Linear Algebra and its Applications*. 1, 103-125, 1968.

Chen, C. and Mangasarian, O. L.; A Class of Smoothing Functions for Nonlinear and Mixed Complementarity Problems. *Computational Optimization and Applications*, 5, 97-138, 1996.

Dirske, S. P. and Ferris, M. C.; The PATH Solver: A Non-Monotone Stabilization Scheme for Mixed Complementarity Problems. *Optimization Methods and Software*, 5, 123-156, 1995a.

Dirske, S. P. and Ferris, M. C.; MCPLIB: A Collection of Nonlinear Mixed Complementarity Problems. *Optimization Methods and Software*, 5, 319-345,1995b.

Ferris, M. C. and Pang, J. S.; Engineering and Economic Applications of Complementarity Problems. Technical Report 95-4, University of Colorado, 1995.

Grossmann, I. E. and Turkay, M.; Solution of Algebraic Systems of Disjunctive Equations. *Comput. Chem. Eng.*, 20:S339–44, 1996. Suppl. Part A.

King, C. J.; Separation Processes, Chemical Engineering Series, 2nd. Edition, McGrawHill, 77-79, 1980.

Lemke, C. E.; Bimatrix Equilibrium Points and Mathematical Programming. *Management Sci.*, 11, 681-689, 1965.

More, J. J.; Global Methods for Nonlinear Complementarity Problems. Technical report, Argonne National Laboratory, Mathematics and Computer Science Division, April 1994.

Westerberg, A. W. and Director, S. W.; A Modified Least Square Algorithm for Solving Sparse n x n Sets of Nonlinear Equations. Technical report, Carnegie Mellon University, Engineering Design Research Center, EDRC-06-5-79, January

1979.

Zaher, J. J.; Conditional Modeling in an Equation-Based Modeling Environment. The Annual AIChE National Meeting, 1991. paper 138c.

Zaher, J. J.; Conditional Modeling. Ph.D. thesis, Department of Chemical Engineering Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1995.

CHAPTER 6 INTERIOR POINT METHODS IN THE SOLUTION OF CONDITIONAL MODELS

Interior point methods have recently become an interesting alternative in a number of numerical applications. In particular, their performance in the solution of problems involving complementarity equations has been the subject of extensive research and their efficacy is well documented. In this chapter, following a description of the fundamentals of interior point methods, we describe the globally convergent framework proposed by Wang *et al* (1996) for solving a constrained system of nonlinear equations by an interior point potential reduction method. Also, we show how we can apply the potential reduction algorithm and its convergence result to the complementarity formulation described in Chapter 5. Based on that observation, we then apply the algorithm proposed by Wang to solve the complementarity examples used as case studies throughout this work. Moreover, we also apply some high order strategies designed to improve convergence (Mehrotra, 1992; Gondzio, 1996), and compare the results obtained with each of the methods. All those techniques have been incorporated to the ASCEND modeling environment with the implementation of the solver IPSlv.

6.1 MOTIVATION

As early as in the late 1940's, almost at the same time as when Dantzig presented the simplex method for linear programming, several researches including Von Neumann (1947) and Frisch (1955) proposed interior point algorithms which transverse across the interior of the feasible region to avoid the complexity of vertex-following algorithms (Andersen *et. al*, 1996). Since then, there is a huge body of literature on the interior point methods which includes surveys and numerous articles. See for example Wright (1997), Lusting *et al.* (1994), and Gondzio and Terlaky (1994). Even though the initial effort on interior point methods was focused on linear programming problems, most recent works have concentrated on the common theoretical foundations of linear and nonlinear programming. It is now widely accepted that interior point methods constitute a powerful tool for solving both very large linear and nonlinear programming problems (Gondzio, 1996).

In the field of chemical engineering, interior point methods have been applied to solve data reconciliation, optimal control, multiperiod design and process optimization problems (Ternet, 1998). Alburquerque *et al*. (1997) and Ternet (1998) employ primal dual interior algorithms and high-order corrections to solve the quadratic programming subproblem within a sequential quadratic programming technique.

In this chapter, we propose the application of interior point techniques for the solution of the complementarity formulation representing a conditional model. This approach is based on the work of Wang *et al*. (1996) and was specially motivated by the works of Simantiraki and Shanno (1995) and Wright and Ralph (1996) which apply infeasible interior point algorithms for mixed complementarity problems. Also, recent results reported in the field of chemical engineering by Ternet (1998) encourage the use of Mehrotra's second order method (Mehrotra,1992) and Gondzio's centrality corrections (Gondzio, 1996) to improve the convergence results obtained by Wang's algorithm.

6.2 BASICS OF PRIMAL-DUAL INTERIOR POINT METHODS

The so called primal-dual interior point methods have been shown to outperform the simplex method on many larger problems and to perform better than other interior point methods (Wright, 1997). In this section, we describe the fundamentals of a primal-dual interior point method. Consider the linear programming problem in standard form:

$$min\{c^T x \mid Ax = b, x \geq 0\} \tag{6.1}$$

The optimally conditions of (6.1) are given by:

$$\begin{aligned} A^T\lambda + s &= c \\ Ax &= b \\ x_i s_i &= 0 \\ x, s &\geq 0 \end{aligned} \tag{6.2}$$

If we modify (6.1) by applying a logarithmic barrier to the variables x then we get:

$$min\left\{c^T x - \tau\sum_i \ln(x_i) \,\middle|\, Ax = b\right\} \tag{6.3}$$

similarly, the optimally conditions of (6.3) are given by:

$$\begin{aligned} A^T\lambda + s &= c \\ Ax &= b \\ XSe &= \tau e \end{aligned} \tag{6.4}$$

where *X* and *S* are the diagonal matrices whose elements are the components of the vectors *x* and *s* correspondingly, and *e* is the vector of all ones.

The system of equations (6.2) can be solved by applying the Newton method and carrying out a linear search to enforce the nonnegativity constraints of *x* and *s*. Unfortunately, we can often take only a small step before the nonnegativity constraints are violated, and, therefore, the iterates make little progress towards the solution. Rather than solving the system of equations (6.2), primal-dual interior point methods focus on the solution of the system (6.4) and introduce the concept of the *central path*. The central path is the set of points which are a solution of (6.4) for $\tau > 0$. The role of the parameter τ is to enforce that all the complementarity products have the same values for all indices *i*. Hence, the central path keeps the iterates biased towards the interior of the nonnegative orthant $(x,s) \geq 0$. Note also that as τ goes to zero, the complementarity products decrease to zero at the same rate. For implementation purposes, τ is defined as the product of the parameters σ and μ, resulting in

$$\begin{aligned} A^T\lambda + s &= c \\ Ax &= b \\ XSe &= \sigma\mu e \end{aligned} \qquad \textbf{(6.5)}$$

μ is generally defined as the complementarity gap (average value of the *n* complementarity products):

$$\mu = \frac{x^T s}{n} \qquad \textbf{(6.6)}$$

and σ is the centering parameter with value between zero and one, $0 \leq \sigma \leq 1$, such that $\sigma = 0$ corresponds to a Newton step and $\sigma = 1$ corresponds to a centering direction in which all the products $x_i s_i$ are equal to μ. Various methods differ in the way that μ and σ are chosen (Wright, 1997).

6.3 HIGH ORDER STRATEGIES FOR INTERIOR POINT METHODS

The solution of a system of equations defined by (6.5) involves the solution of the linearized system

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta\lambda^k \\ \Delta s^k \end{bmatrix} = \begin{bmatrix} c - A^T\lambda^k - s^k \\ b - Ax^k \\ -X^kS^ke + \sigma^k\mu^k e \end{bmatrix} \qquad \textbf{(6.7)}$$

at each iteration. It is estimated that the required factorization of the matrices

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix}$$

takes 60 to 90% of the total CPU time needed to solve a problem (Gondzio, 1994). The use of high order information has been motivated by the idea of using the same factorization for several different solves in order to reduce the number of interior point iterations required. As a matter of fact, second-order methods have been shown to give evident savings over the basic first order methods, and, therefore, they have became the computational state of the art (Lustig *et al.*, 1992). In this section, we will review two of the most extensively applied high order methods found in the interior point literature: Mehrotra's second order method (Mehrotra,1992) and Gondzio's centrality correction (Gondzio, 1994). The algorithmic steps of both of those approaches can be found in Wright (1997) and Ternet (1998).

6.3.1 MEHROTRA'S PREDICTOR-CORRECTOR TECHNIQUE

Mehrotra's predictor-corrector technique (Mehrotra, 1990; Mehrotra, 1992; Lustig *et al.*,1992) has three main components (Wright, 1997; Ternet, 1998):

- A **predictor step**: a pure Newton (also known as affine-scaling) direction. For problem (6.5), this step is calculated by solving (6.7) with $\sigma=0$:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta_p x^k \\ \Delta_p \lambda^k \\ \Delta_p s^k \end{bmatrix} = \begin{bmatrix} c - A^T\lambda^k - s^k \\ b - Ax^k \\ -X^k S^k e \end{bmatrix} \qquad \textbf{(6.8)}$$

- An adaptive approach to compute the centering parameter. This parameter is calculated in terms of the complementarity gap at the current point and the complementarity gap after a hypothetical step in the affine scaling direction is taken. In general, the centering parameter is small when good progress can be made in the affine direction and large when the affine direction produces little improvement. The actual calculation of the centering parameter is given by:

$$\begin{aligned}\mu^k &= (x^k)^T s^k / n \\ \mu_{aff} &= [(x^k + \alpha\Delta_p x^k)^T (s^k + \alpha\Delta_p x^k)]/n \\ \sigma^k &= (\mu^k/\mu_{aff})^3\end{aligned} \qquad \textbf{(6.9)}$$

where the factor α defines the maximum stepsize in the affine-scaling direction that can be taken while preserving the nonnegativity of the variables *(x,s)*.

- A **corrector step:** essentially a step based on Taylor series expansion of the complementarity equations. For problem (6.5), this step is calculated by:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta_c x^k \\ \Delta_c \lambda^k \\ \Delta_c s^k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sigma^k \mu^k e - \Delta_p X^k \Delta_p S^k e \end{bmatrix} \qquad \textbf{(6.10)}$$

where $\Delta_p X^k$ and $\Delta_p S^k$ are the diagonal matrices whose elements are the components of the vectors $\Delta_p x^k$ and $\Delta_p s^k$ calculated in the predictor step.

The corrector step is best understood by the following analysis. If a full step ($\alpha = 1$) were achieved in the affine scaling direction, the new complementarity conditions would be given by:

$$(S^k + \Delta S_p^k)e \cdot (X^k + \Delta X_p^k)e = X^k S^k e + S^k \Delta S_p^k e + X^k \Delta X_p^k e + \Delta X_p^k \Delta S_p^k e \qquad \textbf{(6.11)}$$

However, the last equation of the system of equations (6.8) is:

$$S^k \Delta S_p^k e + X^k \Delta X_p^k e = -X^k S^k e \qquad \textbf{(6.12)}$$

and (6.11) reduces to:

$$(S^k + \Delta S_p^k)e \cdot (X^k + \Delta X_p^k)e = \Delta X_p^k \Delta S_p^k e \qquad \textbf{(6.13)}$$

Hence, the second order correction $\Delta_p X^k \Delta_p S^k e$ corresponds to the violation of the complementarity conditions if a full step in the affine-scaling direction were taken. Moreover, the corrector direction is supposed to drive from a hypothetical point $(x^k + \Delta x_p^k, s^k + \Delta s_p^k, \lambda^k + \Delta\lambda_p^k)$ to a point in the central trajectory (note also the presence of the centering correction $\sigma^k \mu^k e$ in the corrector step).

Because the corrector step is a second order correction, the stepsize determination in Mehrotra's technique consists of finding an $\bar{\alpha} > 0$ such that for all $\alpha \in (0, \bar{\alpha})$ (Wright, 1997):

$$\begin{gathered} (x, s, \lambda)^{k+1} = (x, s, \lambda)^k + \alpha \cdot \Delta_p^k + \alpha^2 \Delta_c^k \\ (x, s)^{k+1} \geq 0 \end{gathered} \tag{6.14}$$

where $\Delta_p{}^k$ and $\Delta_c{}^k$ are the predictor and the corrector steps in the variables correspondingly.

6.3.2 GONDZIO'S CENTRALITY CORRECTIONS

Gondzio (Gondzio,1994; Gondzio and Terlaky, 1994; Andersen *et al.*, 1996) argues that what reduces most the efficiency of an interior point method is a large discrepancy among the complementarity products. Complementarity products that are too small or too large (compared with the average) are undesirable, with the former being the more disastrous.

Gondzio assumes that a predictor step (a step calculated with Mehrotra's approach, for instance) has been calculated and evaluates the complementarity products for the current point (trial point) $(\tilde{x}, \tilde{s})$:

$$\tilde{\vartheta} = (\tilde{X}\tilde{S}e) \in R^n \tag{6.15}$$

Then, the trial point is projected componentwise onto a hypercube:

$$Hyp = [\beta_{min}\mu^k, \beta_{max}\mu^k]^n \tag{6.16}$$

to define the target

$$\vartheta_t = \pi(\tilde{\vartheta} | Hyp) \in R^n \tag{6.17}$$

Gondzio believes that the effort should be concentrated on correcting only outlier complementarity products (the ones that do not belong to the interval $[\beta_{min}\mu^k, \beta_{max}\mu^k]$). The parameters β_{min} and β_{max} in (6.16) are the relative threshold values for outlier complementarity products and are assumed as given (in Gondzio's implementation $\beta_{min}=0.1$ and $\beta_{max}=10$).

Hence, a corrector term of the current iterate solves the system:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta_m x^k \\ \Delta_m \lambda^k \\ \Delta_m s^k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vartheta_t - \tilde{\vartheta} \end{bmatrix} \tag{6.18}$$

where $\vartheta_t - \tilde{\vartheta}$ is given by (Wright, 1997;Ternet, 1998):

$$(\vartheta_t - \tilde{\vartheta})_i = \begin{cases} (\beta_{min}\mu^k - \tilde{\vartheta}_i) & if\, \tilde{\vartheta}_i < \beta_{min}\mu^k \\ (\beta_{max}\mu^k - \tilde{\vartheta}_i) & if\, \tilde{\vartheta}_i > \beta_{max}\mu^k \\ -\beta_{max}\mu^k & if\, \tilde{\vartheta}_i > 2\beta_{max}\mu^k \\ 0 & otherwise \end{cases} \tag{6.19}$$
$$\forall i \in 1 \ldots n$$

Note that system (6.18) is full of zeros since $\vartheta_t - \tilde{\vartheta}$ is nonzero only for components that refer to the complementarity products that do not belong to $[\beta_{min}\mu^k, \beta_{max}\mu^k]$. Also note that for large complementarity products, $\tilde{\vartheta}_i > 2\beta_{max}\mu^k$, a more conservative value for the target is used in order to prevent the undesirable effect of bad scaling (Wright, 1997).

6.4 AN INTERIOR POINT POTENTIAL REDUCTION METHOD FOR CONSTRAINED EQUATIONS

In this section we describe the globally convergent framework developed by Wang *et al.* (1996). They combined the classical damped Newton Method with interior point potential reduction methods for solving a constrained system of nonlinear equations. They show how their formulation provides a unified framework for many mathematical problems, including complementarity problems and nonlinear programs.

6.4.1 THE MODEL

Let Ω be a closed subset of the Euclidean space R^n , where n is a positive integer. Ω is assumed to have a nonempty interior, denoted by *int* Ω. Let $H{:}\Omega \to R^n$ be a continuous mapping defined on Ω. Consider the problem of finding a vector ω satisfying:

$$H(\omega) = 0 \qquad \textbf{(6.20)}$$

The basic structure that Wang *et al*. impose is that there is a partition of the range space of the mapping H :

$$H(\omega) = \begin{bmatrix} F(\omega) \\ G(\omega) \end{bmatrix}, \omega \in \Omega \qquad \textbf{(6.21)}$$

for some functions $F{:}\Omega \to R^{n_1}$ and $G{:}\Omega \to R^{n_2}$ with $n_1 + n_2 = n$. Now let

$$\Omega_\Sigma = \{\omega \in int\Omega | G(\omega) > 0\} \qquad \textbf{(6.22)}$$

where the inequality $G(\omega)>0$ is to be understood component wise. The assumptions made by Wang and coworkers are the following:

1. $\Omega_\Sigma \neq \emptyset$ and $\Omega_\Sigma = \{\omega \in \Omega | G(\omega) > 0\}$.
2. H is continuously differentiable on the set Ω_Σ.
3. The Jacobian matrix $H'(\omega)$ is nonsingular for all $\omega \in \Omega_\Sigma$.

A real-valued function $\psi(\omega){:}\ \Omega_\Sigma \to R$ is also defined:

$$\psi(\omega) = \zeta \cdot \log(F(\omega)^T F(\omega) + e^T G(\omega)) - \sum_i^{n_2} \log G_i(\omega) \qquad \textbf{(6.23)}$$

where ζ is a fixed but arbitrary scalar satisfying $\zeta > n_2$ and e is the vector of all ones. When $n_2=0$, (6.23) reduces (without the logarithm) to the standard merit function used in the Newton method for solving the system of equations $F(\omega)=0$. If we decrease $\psi(\omega)$, the first term in (6.23) contributes to reduce some norm of the function *H(*ω*)*. The second term is the logarithmic barrier function whose role is to prevent a point ω from reaching the boundary of Ω_Σ.

Hence, the goal of Wang's approach is to develop an iterative algorithm for solving (6.21) while (6.23) plays the role of a merit function which guides the generation of the iterates ω^k.

6.4.2 AN INTERIOR POINT APPROACH

Wang's approach originates from the extension of path-following interior point methods for their application to the nonlinear case. The algorithm proposes to calculate a perturbed Newton direction in order to keep the iterates close to the central path of Ω_Σ. For a given vector $\omega \in \Omega_\Sigma$, the perturbed Newton direction at ω is obtained by solving the following system of linear equations:

$$H'(\omega)d = \begin{bmatrix} F'(\omega)d \\ G'(\omega)d \end{bmatrix} = \begin{bmatrix} -F(\omega) \\ -G(\omega) + \sigma\mu(\omega)e \end{bmatrix} \qquad \textbf{(6.24)}$$

where $\mu(\omega) = e^T G(\omega) / n_2$, and σ is the centering parameter. Based on the work of Kojima *et al.* (1994), Wang shows that a solution *d* of (6.24) is a descent direction for the function $\psi(\omega)$ defined by (6.23) such that:

$$\psi(\omega + \lambda d) - \psi(\omega) \le -\alpha\lambda(1 - \sigma)(\zeta - n_2) \qquad \textbf{(6.25)}$$

where $\alpha \in (0,1)$ and λ is a scalar such that for all $\lambda \in (0,\bar{\lambda})$, $\bar{\lambda}>0$, we get $\omega + \lambda d \in \Omega_\Sigma$ (*i.e.* $\bar{\lambda}$ is the factor in the linear search of the damped Newton method). Note that the right hand side of (6.25) is always negative.

Furthermore, in order to prove convergence, Wang and coworkers define a function $\nu(\omega)$ as a measure of the error of a given iterate ω^k:

$$\nu(\omega) = F(\omega)^T F(\omega) + e^T G(\omega) \qquad \textbf{(6.26)}$$

and show that a sequence of iterates generated by their approach satisfies the following properties:

1. The sequence $\{\nu(\omega^k)\}$ is bounded.
2. If the sequence of iterates $\{\omega^k\}$ is bounded, then

$$\lim_{k \to \infty} \nu(\omega^k) = 0 \tag{6.27}$$

The derivation of equation (6.25) is shown in Appendix E. For the derivation of equation (6.27) and the demonstration of the above properties, the reader is referred to the paper of Wang *et al*. (1996).

6.4.3 THE ALGORITHM

The steps of the potential reduction algorithm for solving (6.21) are:

Step 1. Initialization. Let $\varsigma > n_2$, $\alpha \in (0,1)$, $\bar{\sigma} \in [0, 1)$, $\xi > 0$ and $\rho \in (0,1)$ be given. Choose a $\sigma^0 \in [0,\bar{\sigma}]$ and an initial point $\omega^0 \in \Omega_\Sigma$.

Step 2. Direction Generation. Solve the system of linear equations (6.24) at $\omega=\omega^k$:

$$H'(\omega^k)d = \begin{bmatrix} F'(\omega^k)d \\ G'(\omega^k)d \end{bmatrix} = \begin{bmatrix} -F(\omega^k) \\ -G(\omega^k) + \sigma^k\mu(\omega^k)e \end{bmatrix}$$

where $\mu(\omega^k) = e^T G(\omega^k)/n_2$, to obtain the search direction d^k.

Step 3. Stepsize determination. Let m_k be the smallest nonnegative integer m such that the following conditions hold:

$$\omega^k + \xi\rho^m d^k \in \Omega_\Sigma \tag{6.28}$$

$$\psi(\omega^k + \xi\rho^m d^k) - \psi(\omega^k) \le -\alpha\xi\rho^m(1 - \sigma^k)(\zeta - n_2) \tag{6.29}$$

and set $\omega^{k+1} = \omega^k + \xi\rho^{m_k} d^k$.

Step 4. Termination check. Terminate if ω^{k+1} satisfies a prescribed stopping rule. Otherwise, pick a $\sigma^{k+1} \in [0,\bar{\sigma}]$ and return to **Step 2**.

The values suggested by Wang for each of the parameters defined in **Step 1** are: $\varsigma = n_2 \cdot \sqrt{n_2}$, $\alpha = 0.5$, $\sigma = 0.5$, $\xi = 2$ and $\rho = 0.5$.

6.5 THE COMPLEMENTARITY REPRESENTATION OF A CONDITIONAL MODEL AND ITS RELATION TO WANG'S FRAMEWORK

Consider the complementarity formulation described by (5.17) for a system of equations containing a disjunction with two disjunctive terms:

$$h(x) = 0$$

$$\begin{pmatrix} r_{1_j}(x) - p_{1_j} \\ r_{2_j}(x) - p_{2_j} \\ g_{l-\beta}(x) - p_{1_l} + p_{2_l} \end{pmatrix} = 0 \qquad \forall j \in [1\ldots\beta], l \in [\beta+1\ldots\beta+\gamma]$$

$$\sum_{t=1}^{\beta+\gamma-s} p_{1_t} \cdot p_{2_{t+s}} + \sum_{t=\beta+\gamma-s+1}^{\beta+\gamma} p_{1_t} \cdot p_{2_{t+s-\beta-\gamma}} = 0 \qquad \forall s \in [0\ldots\beta+\gamma-1]$$

$$p_{i_q} \ge 0 \qquad \forall i \in [1\ldots2], q \in [1\ldots\beta+\gamma]$$

and let n_x be the dimensionality of the vector of variables of the original conditional model, *i.e.* $x \in R^{n_x}$. The resulting complementarity problem (5.17) is one of $n_x + 2(\beta+\gamma)$. variables and equations. Recall also that the positiveness in the variables p_{i_q} is imposed, that is $p_{i_q} \in R_+^{2(\beta+\gamma)}$. If we define:

- $\Omega = R_+^{2(\beta+\gamma)} \times R^{n_x}$
- $\omega \in \Omega$, that is $\omega = (p_{i_q}, x) \in R_+^{2(\beta+\gamma)} \times R^{n_x}$
- the partition $H(\omega)$ given by:

$$F(\omega)=\begin{pmatrix} \underline{h}(x) \\ r_{1_j}(x)-p_{1_j} \\ r_{2_j}(x)-p_{2_j} \\ g_{l-\beta}(x)-p_{1_l}+p_{2_l} \end{pmatrix}=0 \qquad \forall j\in[1\ldots\beta],\, l\in[\beta+1\ldots\beta+\gamma]$$

(6.30)

$$G(\omega)=\left(\sum_{t=1}^{\beta+\gamma-s} p_{1_t}\cdot p_{2_{t+s}}+\sum_{t=\beta+\gamma-s+1}^{\beta+\gamma} p_{1_t}\cdot p_{2_{t+s-\beta-\gamma}}\right)=0 \qquad \forall s\in[0\ldots\beta+\gamma-1]$$

and assume:

- the continuous differentiability of the function $F(\omega)\colon R_+^{2(\beta+\gamma)}\times R^{n_x}$ at every point $\omega=(p_{i_q},x)\in R_{++}^{2(\beta+\gamma)}\times R^{n_x}$.
- and the nonsingularity of Jacobian matrix $H'(\omega)$ (this also implies that we assume the nondegeneracy of the complementarity equations $G(\omega)$ in order to avoid numerical singularities caused by such equations)

then we can apply the potential reduction algorithm and its convergence result to the complementarity formulation described earlier in this work.

6.6 IMPLEMENTATION OF WANG'S ALGORITHM AND HIGH ORDER TECHNIQUES FOR SOLVING A CONDITIONAL MODEL

The algorithm proposed by Wang and the second order corrections developed by Mehrotra and Gondzio have been implemented in a computer program called IPSlv for their application to the solution of conditional models formulated as a complementarity problem of the type represented by (6.30). The solver IPSlv has been incorporated into the ASCEND environment. Some details of the implementation of Wang's algorithm and both of the second order techniques are given in this section.

6.6.1 PRACTICAL IMPLEMENTATION OF WANG'S ALGORITHM

For purposes of implementation, we made some minor modifications to the algorithm described in 6.4.3:

1. First, the value for the centering parameter σ in Wang's algorithm is rather arbitrary. Wang's proposes using a value of 0.5 and dividing that value by 10 every time that a full Newton step in the variables can be taken. In our implementation we provide the option of defining an initial value of the centering parameter and modifying it under Wang's heuristic. However, we also support an evaluation of the centering parameter *a la* Mehrotra. For the later case, we need to substitute **Step 2** (direction generation) in algorithm 6.4.3 by the following steps:

   - Calculate the affine direction d_p^k by solving (6.24) with $\sigma = 0$.

$$H'(\omega^k)d_p^k = \begin{bmatrix} F'(\omega^k)d_p^k \\ G'(\omega^k)d_p^k \end{bmatrix} = \begin{bmatrix} -F(\omega^k) \\ -G(\omega^k) \end{bmatrix} \qquad \textbf{(6.31)}$$

   - Calculate the centering parameter (similarly to (6.9)):

$$\begin{aligned} \mu^k(\omega^k) &= e^T G(\omega^k)/n_2 \\ \mu_{aff}(\omega^k) &= e^T G(\omega^k + \lambda d_p^k)/n_2 \\ \sigma^k &= [\mu^k(\omega^k)/\mu_{aff}(\omega^k)]^3 \end{aligned} \qquad \textbf{(6.32)}$$

   where is λ is a scalar such that for all $\lambda \in (0,\bar{\lambda})$, $\bar{\lambda}>0$, we get $\omega^k + \lambda d_p^k \in R_+^{2(\beta+\gamma)} \times R^{n_x}$ (*i.e.* $\bar{\lambda}$ is the factor in the linear search of the damped Newton method).

   - Solve equation (6.24) with the centering parameter calculated in the previous step.

$$H'(\omega^k)d_c^k = \begin{bmatrix} F'(\omega^k)d_c^k \\ G'(\omega^k)d_c^k \end{bmatrix} = \begin{bmatrix} -F(\omega^k) \\ -G(\omega^k) + \sigma^k \mu(\omega^k) e \end{bmatrix} \qquad \textbf{(6.33)}$$

   Here, the direction d_c^k is then used in **Step 3** (stepsize determination) of the algorithm

6.4.3. Note that, even though two solves are required, the factorization of the Jacobian matrix $H'(\omega)$ is being done only once (as in second order methods). In other words, this option requires an extra backsolve to compute the affine direction but leaves us with a more reliable value for the centering parameter σ^k.

2. The second modification arises from the fact that, while solving a conditional model as a complementarity problem, the equations of the partition $G(\omega)$ may contain the summation of complementarity products rather than individual complementarity products. For that reason, we modified the second term in the right hand side (the logarithmic barrier term) of the merit function $\psi(\omega)$, Equation (6.23), to include the summation of the logarithm of each individual complementarity product instead of the logarithm of each function $G_i(\omega)$. That modification is motivated solely by the observation that we need to prevent each individual complementarity product from going prematurely to zero (a $G_i(\omega)$ could be greater than zero but still an individual complementarity product in it could reach zero), but there is presently no theoretical basis to support it. As a matter of fact, Wang's proof of convergence does not strictly apply anymore after this modification is made.

6.6.2 IMPLEMENTATION OF SECOND ORDER CORRECTIONS

The linearization of the functions $F(\omega)$ in (6.24) may lead to infeasibilities in the solution to the problem (6.21). Hence, in order to strictly apply second order corrections to the problem (6.24), we should also consider a correction for the linearization of the equations $F(\omega)$. Still, here we are mainly concerned with the numerical problems associated with the complementarity equations $G(\omega)$, and, for those equations, an analogy to Mehrotra's second order correction and Gondzio's centrality correction can be derived. In our implementation of second order techniques, the main difference with respect to the original approaches is that, in the problem we are solving, the equations of the partition $G(\omega)$ may contain the summation of complementarity products.

6.6.2.1 Applying Mehrotra's corrector step

As in our implementation of Wang's algorithm, the implementation of Mehrotra's predictor corrector technique implies a modification of the direction generation step (**Step**

2) of algorithm 6.4.3. The three elements of Mehrotra's technique applied to the framework used in this work are:

- **Predictor** step. Calculation of the affine direction as in (6.31).
- Computation of the centering parameter σ^k as in (6.32).
- **Corrector** step. A second order corrector step for the system defined by (6.24) is given by the solution to the linear system:

$$H'(\omega^k)d_c^k = \begin{bmatrix} F'(\omega^k)d_c^k \\ G'(\omega^k)d_c^k \end{bmatrix} = \begin{bmatrix} 0 \\ \sigma\mu(\omega^k)e - C(d_p^k) \end{bmatrix} \qquad \textbf{(6.34)}$$

where the vector $C(d_p^k)$ is a function of the affine direction and represents the second order correction for the partition *G*(ω) (similar to the second order correction $\Delta X_p{}^k \Delta S_p{}^k e$ used in (6.10)). Specifically for the complementarity formulation used in this work (Equation (6.30)), the elements of the vector $C(d_p^k)$ are given by:

$$C_{s+1}(d_p^k) = \sum_{t=1}^{\beta+\gamma-s} \Delta p_{1_t} \cdot \Delta p_{2_{t+s}} + \sum_{t=\beta+\gamma-s+1}^{\beta+\gamma} \Delta p_{1_t} \cdot \Delta p_{2_{t+s-\beta-\gamma}} \qquad \textbf{(6.35)}$$
$$\forall s \in [0 \ldots \beta+\gamma-1]$$

For simplicity in the representation, note that we have omitted the index *k* (iteration) for the complementary variables p_{i_q}. Also, note that the corrector term for the partition *F*(ω) in (6.34) is zero. As stated before, a value of zero is strictly correct only in a case in which the equations of *F*(ω) are linear.

As an analogy to the discussion about the meaning of Mehrotra's second order correction given before for the linear programming case, the following simple example illustrates the meaning of the second order correction defined by (6.35):

Consider the complementarity equation (including two complementarity products) given by:

$$G_1(\omega) = 0 \qquad \Longrightarrow \qquad p_{1_1} \cdot p_{2_1} + p_{1_2} \cdot p_{2_2} = 0$$

The linearization of that complementarity equation in a Newton scheme is given by:

$$G_1'(\omega)d = -G(\omega) \qquad \Longrightarrow$$

$$p_{1_1} \cdot \Delta p_{2_1} + \Delta p_{1_1} \cdot p_{2_1} + p_{1_2} \cdot \Delta p_{2_2} + \Delta p_{1_2} \cdot p_{2_2} = -(p_{1_1} \cdot p_{2_1} + p_{1_2} \cdot p_{2_2}) \qquad \textbf{(6.36)}$$

If a full step were achieved in the affine-scaling (Newton) direction, the new complementarity conditions would be given by:

$$(p_{1_1} + \Delta p_{1_1}) \cdot (p_{2_1} + \Delta p_{2_1}) + (p_{1_2} + \Delta p_{1_2}) \cdot (p_{2_2} + \Delta p_{2_2}) = p_{1_1} \cdot p_{2_1} + p_{1_2} \cdot p_{2_2} + p_{1_1} \cdot \Delta p_{2_1} + \Delta p_{1_1} \cdot p_{2_1} + p_{1_2} \cdot \Delta p_{2_2} + \Delta p_{1_2} \cdot p_{2_2} + \Delta p_{1_1} \cdot \Delta p_{2_1} + \Delta p_{1_2} \cdot \Delta p_{2_2}$$

which, after substituting (6.36), reduces to

$$(p_1^1 + \Delta p_1^1) \cdot (p_1^2 + \Delta p_1^2) + (p_2^1 + \Delta p_2^1) \cdot (p_2^2 + \Delta p_2^2) = \Delta p_{1_1} \cdot \Delta p_{2_1} + \Delta p_{1_2} \cdot \Delta p_{2_2} \qquad \textbf{(6.37)}$$

The right hand side of (6.37) is the second order corrector term defined by (6.35). Similar to the linear case, the second order correction (6.35) corresponds to the violation of the complementarity conditions *G(*ω*)* if a full step in the affine-scaling direction were taken. Hence, the corrector direction is supposed to drive from a hypothetical point ($\omega^k + d_p^k$) to a point in the central trajectory.

When applying a second order correction, we also have to modify **Step 3** of algorithm 6.4.3 (stepsize determination). Since the corrector step is a second order term, equation (6.28) has to be substituted by

$$\omega^k + \xi\rho^m d_p^k + (\xi\rho^m)^2 d_c^k \in \Omega_\Sigma \qquad \textbf{(6.38)}$$

where d_p^k is the affine direction (predictor step) and d_c^k is the second order corrector direction. Recall that, specifically for our problem, $\Omega = R_+^{2(\beta+\gamma)} \times R^{n_x}$ and $\Omega_\Sigma = R_{++}^{2(\beta+\gamma)} \times R^{n_x}$.

6.6.2.2 Applying Gondzio's centrality corrections

As in Gondzio's original method, in this implementation we assume that a predictor direction (a Mehrotra's step or a step generated by Step 2 in Wang's algorithm, for instance) has been calculated, and, therefore, the complementarity equations for the resulting point can be evaluated:

$$\tilde{\Upsilon} = G(\tilde{\omega}) \in R^{n_2} \quad \textbf{(6.39)}$$

If we would strictly follow Gondzio's method, the next step would be to define a hypercube

$$Hyp = [\beta_{min}\mu^k, \beta_{max}\mu^k]^{n_2} \quad \textbf{(6.40)}$$

and project the complementarity equations componentwise to define the target:

$$\Upsilon_t = \pi(\tilde{\Upsilon} | Hyp) \in R^{n_2} \quad \textbf{(6.41)}$$

In Gondzio's original approach, each complementarity equation contains only one complementarity product. In that way, a correction term can be evaluated for each complementarity product. On the contrary, here we recall once again the fact that, for the complementarity representation of a conditional models, the complementarity equations $G(\omega)$ may consist of the summation of complementarity products rather than individual complementarity products. Hence, the value of a corrector term for each complementarity product cannot be explicitly estimated.

In this implementation of Gondzio's centrality correction, equations(6.40) and (6.41) are not used directly. In order to best describe our implementation of Gondzio's centrality correction, let n_g be the number of complementarity products in each complementarity equation, and for purposes of illustration, consider that n_g is the same for each complementarity equation.

Hence, the complementarity equations evaluated at the trial point can be represented in terms of the individual complementarity products as:

$$G_i(\tilde{\omega}) = \sum_{j=1}^{j=n_g} g_{ij}(\tilde{\omega}) \qquad \forall i \in 1 \ldots n_2 \tag{6.42}$$

where $g_{ij}(\tilde{\omega})$ is the *j-th* complementarity product in the *i-th* complementarity equation. Furthermore, the components of a vector $\tilde{\gamma} \in R^{n_2 \times n_g}$ including all the individual complementarity products are given by:

$$\tilde{\gamma}_{(i-1) \cdot n_g + j} = g_{ij}(\tilde{\omega}) \qquad \forall i \in 1 \ldots n_2, \forall j \in 1 \ldots n_g \tag{6.43}$$

Once the vector of individual complementarity products for the trial point $\tilde{\gamma}$ has been evaluated, we can use a procedure analogous to the one suggested by Gondzio. The vector $\tilde{\gamma}$ is projected on a hypercube

$$Hyp_g = [\beta_{min}(\mu^k / n_g), \beta_{max}(\mu^k / n_g)]^{n_2 \times n_g} \tag{6.44}$$

to define the target

$$\gamma = \pi(\tilde{\gamma} | Hyp_g) \in R^{n_2 \times n_g} \tag{6.45}$$

which results in the difference vector $\gamma - \tilde{\gamma}$ being:

$$(\gamma - \tilde{\gamma})_k = \begin{cases} (\beta_{min}(\mu^k / n_g) - \tilde{\gamma}_k) & if \tilde{\gamma}_k < \beta_{min}(\mu^k / n_g) \\ (\beta_{max}(\mu^k / n_g) - \tilde{\gamma}_k) & if \tilde{\gamma}_k > \beta_{max}(\mu^k / n_g) \\ -\beta_{max}(\mu^k / n_g) & if \tilde{\gamma}_k > 2\beta_{max}(\mu^k / n_g) \\ 0 & otherwise \end{cases} \tag{6.46}$$

$$\forall k \in 1 \ldots n_2 \times n_g$$

Thus, we define the corrector term for each complementarity equation as the summation of the corrector terms for each of the complementarity products:

$$C_i(\tilde{\omega}) = \sum_{k=(i-1)\cdot n_g+1}^{k=i\cdot n_g} (\gamma - \tilde{\gamma})_k \qquad \forall i \in 1\ldots n_2 \tag{6.47}$$

Finally, an analogy to Gondzio's centrality correction for the complementarity representation of a conditional model will be given by:

$$H'(\omega^k)d_g^k = \begin{bmatrix} F'(\omega^k)d_g^k \\ G'(\omega^k)d_g^k \end{bmatrix} = \begin{bmatrix} 0 \\ C(\tilde{\omega}) \end{bmatrix} \tag{6.48}$$

To summarize, in our implementation of Gondzio's method,

- the complementarity equations are decomposed in terms of their individual complementarity products,
- a correction term for each outlier complementarity product is evaluated,
- the corrector term for a complementarity equation is obtained as the summation of the corrections of each of its individual complementarity products and, finally,
- Gondzio's corrector step is evaluated by solving the linear system defined by (6.48).

This implementation was motivated by Gondzio's original goal of correcting (if necessary) each complementarity product of each complementarity equation. However, we recognize here a fundamental incorrectness of the approach presented in this section: there is no guarantee that the corrector term $C_i(\tilde{\omega})$ indeed provides a correction for each complementarity product. Hence, this implementation is not theoretically sound, and it should only be considered as an initial study to discover the difficulties and the benefits of the implementation on the solution of our complementarity formulation. Future work should be aimed for obtaining a theoretically sound implementation.

6.6.3 THE SOLVER IPSLV

The numerical techniques described in this section have been implemented in a computer code called IPSlv and incorporated to the ASCEND modeling environment. As such, the selection of the specific technique as well as the values of the various parameters can be specified interactively through the ASCEND solver interface. This implementation can be applied not only to the complementarity representation of a conditional model but also to any kind of complementarity problems described by (6.21). Also, it can be used for any system like (6.5), generated by the optimality conditions of a linear programming problem. The main options offered by this solver are:

1. Numerical technique:
   - Wang's potential reduction method.
   - Mehrotra's second order method.

   Gondzio's centrality corrections can be applied to the point generated by any of those techniques at each iteration.

2. Calculation of the centering parameter σ:
   - Given initial value modified heuristically as proposed by Wang.
   - Calculated *a la* Mehrotra by using the affine scaling direction.

   If Mehrotra's technique is being used, only the second alternative is available. Calculation *a la* Mehrotra is the default option.

3. Calculation of the potential function $\psi(\omega)$:
   - Include the summation of residuals of the complementarity equations in the logarithmic barrier term.
   - Include the summation of the values of the individual complementarity products in the logarithmic barrier term. This is the default option as described earlier.

6.7 NUMERICAL RESULTS

The numerical methods described in this chapter were used to solve the complementarity problems described in Chapter 5. The number of iterations that we used to obtain the solution of each of these problems is shown in Table 6-1.

Table 6-1 Numerical results by using interior point methods.

Example	Reference	Wang	Wang + Mehrotra	Wang + Gondzio	Wang + Mehrotra+ Gondzio
Flow Transition (sonic-subsonic)	Zaher (1995)	7	7	7	7
Phase Equilibria	Zaher (1995)	11	8	11	8
Heat exchanger	Zaher (1995)	9	8	9	8
Pipeline network	Bullard and Biegler (1992)	34	33	36	33
Simple L-V flash	King (1980)	26	23	32	23
Linear mass balance	Grossmann and Turkay (1996)	15	14	15	14

6.8 DISCUSSION

Regarding the numerical results obtained for this set of examples, we can make the following observations:

- A sequence of steps generated by Wang's algorithm successively converged to a the solution in all of the complementarity problems.
- Also, in all the cases, the use of Mehrotra's second order correction resulted in a reduction in the number of iterations required for convergence by Wang's algorithm.
- Finally, the application of Gondzio's corrector step did not provide a significant improvement with respect to a step calculated solely by Wang's algorithm or a step including Mehrotra's correction.

However, we are aware that this reduced set of examples may not be enough to draw a meaningful conclusion regarding the numerical performance of each of the techniques. So, for instance, we still consider that a corrector step a la Gondzio might reduce the problems associated with scaling in some other complementarity problems.

Rather than a formal comparison among the performance of these methods, we consider that a major result obtained from the experiments shown in the previous section is that we could establish the viability of the use of interior point methods in the solution of conditional models represented as complementarity problems. That is what we believe is one the contributions of this work.

The use of interior point methods in the solution of conditional models represented as complementarity problems provides some theoretical and numerical advantages with respect to both of the approaches discussed in section 5.4 of Chapter 5:

1. First, a theoretical proof of convergence exists for the interior point algorithm used in this work (Wang *et al.*, 1996).
2. The use of interior point methods will prevent the complementarity products of the complementarity equations from prematurely reaching zero. In practice, this feature benefits the numerical performance by *i)* avoiding numerical singularities during the iterative solution process and *ii)* allowing to take larger steps in a Newton-based technique before the nonnegativity constraints of the complementarity variables are violated.
3. The use of second order corrections in interior point methods, specially Gondzio's centrality correction, can help prevent numerical problems associated to bad scaling.

Finally, the implementation of second order techniques for the complementarity formulation developed in this work must only be considered as an initial study which discovers the complications and potential benefits of such an implementation. Hence, we want to emphasize that a conceptual gap still exists between the application of second order methods for the linear programming problem case and their application for our complementarity formulation; future research should be aimed at bridging this gap as far as possible.

6.9 REFERENCES

Albuquerque, J. S., Gopal, V., Staus, G., Biegler, L. T. and Ydstie, B. E.; Interior Point SQP Strategies for Structured Process Optimization Problems. to appear in *Comput. Chem. Eng*, 1997.

Andersen, E. D., Gondzio, J., Meszaros, C. and Xu, X.; Implementation of Interior Point Methods for Large Scale Linear Programming. Technical Report 1996.3, Logilab, University of Geneva, Switzerland, 1996.

Bullard, L.G. and Biegler, L.T.; Iterated Linear Programming Strategies for Nonsmooth Simulation: Continuous and Mixed-Integer Approach. *Comput. Chem. Eng.*, 16(10), 1992.

Frisch, K. R.; The Logarithmic Potential Method of Convex Programming. Technical Report, University Institute of Economics, Oslo, Norway, 1955.

Gondzio, J.; Multiple Centrality Corrections in a Primal-Dual Method for Linear Programming. Computational Optimization and Applications, 6, 137-156, 1996.

Gondzio, J. and Terlaky, T.; A Computational View of Interior-Point Methods for Linear Programming. Technical Report 1994.22, Logilab, University of Geneva, Switzerland, 1994.

Grossmann, I. E. and Turkay, M.; Solution of Algebraic Systems of Disjunctive Equations. *Comput. Chem. Eng.*, 20, S339–44, 1996. Suppl. Part A.

Jansen, B., Roos, C., Terlaky, T. and Vial, J.; Primal Dual Target Following Algorithms for Linear Programming. Technical Report 93-107, Faculty of Technical Mathematics and Informatics, Delft University of Technology, The Netherlands, 1993.

King, C. J.; Separation Processes, Chemical Engineering Series, 2nd. Edition, McGrawHill, 77-79, 1980.

Kojima, M., Noma, T. and Yoshise, A.; Global Convergence in Infeasible-Interior-Point Algorithms. *Mathematical Programming*, 65, 43-72, 1994.

Lustig, I. J., Marsten, R. E. and Shanno, F.; On Implementing Mehrotra’s Predictor-Corrector Interior-Point Method for Linear Programming. *SIAM Journal of Optimization*. 2(3). 435-449, 1992.

Lustig, I. J., Marsten, R. E. and Shanno, F.; Interior Point Methods for Linear Programming: Computational State of the Art. *ORSA Journal for Computing*. 6, 1-14, 1994.

Mehrotra, S.; Higher Order Methods and their Performance. Technical Report 90-16R1, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 1991.

Mehrotra, S.; On the Implementation of a Primal-dual Interior Point Method. *SIAM Journal of Optimization*. 2(4), 575-601, 1992.

Simantiraki, E. M., Shanno, D. F.; An Infeasible-Interior-Point Algorithm for Solving Mixed Complementarity Problems. Rutcor Research Report 37-95., Rutgers Center for Operations Research, Rutgers University, 1995.

Ternet, D. J.; New Approaches to a Reduced Hessian Successive Quadratic Programming Method for Large-Scale Process Optimization. Ph.D. thesis, Department of Chemical Engineering Carnegie Mellon University, Pittsburgh, Pennsylvania, April, 1998.

Von Newman, J.; On a Maximization Problem. Technical Report, Institute for Advanced Study, Princeton, NJ, 1947.

Wang, T., Monteiro, R. D. C., and Pang, J.; An Interior Point Potential Reduction Method for Constrained Equations. *Mathematical Programming*, 74, 159-195, 1996.

Wright, S.; An Interior Point Algorithm for Linearly Constrained Optimization. *SIAM Journal of Optimization*. 1(4), 1992.

Wright, S.; Primal Dual Interior Point Methods. First Edition. SIAM press, Philadelphia, PA, 1997.

Wright, S. and Ralph, D.; A Superlinear Infeasible-Interior-Point Algorithm for Monotone Complementarity Problems. *Mathematics of Operation Research*, 21, 815-838, 1996.

Zaher, J. J.; Conditional Modeling. Ph.D. thesis, Department of Chemical Engineering Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1995.

Chapter 7 Contributions and Future Work

This chapter concludes the thesis by giving a summary of the work and contributions made in this research. Also directions and recommendations for future work are highlighted.

7.1 SUMMARY

One of the goals of our research group has been to improve one's ability to develop and solve process models. This work was specially focused on the issues involved in the modeling, formulation, structural analysis, and solution of conditional models in an equation-based environment.

In Chapter 2 we identified the modeling capabilities needed for an efficient representation of conditional models. We then described modeling tools for the performance of each of the identified tasks and gave the details of the computer implementation of these tools. Several chemical engineering examples were used to demonstrate the scope of application of the proposed extensions and to show how the expressiveness of an equation-based modeling language increases with their incorporation.

In Chapter 3 we introduced an extension to Zaher's methodology (Zaher, 1993; Zaher, 1995) for the structural analysis of conditional models. This extension allows the consistency analysis to be applied to conditional models in which the number of variables and equations for each of the alternatives may not be the same. Also, we show how, by taking advantage of the structure of the problem, it is sometimes possible to reduce the effort required by such an analysis. In particular, the cases of the existence of repeated structures and common incidence pattern among alternatives were discussed.

In Chapter 4 we investigated the solving of conditional models using a boundary crossing algorithm proposed by Zaher. This algorithm involves the execution of several well differentiated activities including logical analysis, continuous reconfiguration of the equations constituting the problem, calculation of Newton-like steps, and the calculation of subgradient steps. We described the practical implementation of the boundary crossing algorithm as a conditional modeling solution tool and solved several examples of conditional models in chemical engineering.

In Chapter 5 we described a complementarity formulation for representing algebraic systems of disjunctive equations. We show that this formulation not only establishes the complementarity condition among equations belonging to different disjunctive terms but also enforces simultaneous satisfaction of all of the equations appearing within the same disjunctive term. The result of applying our formulation to a disjunctive set of equations is a square system of nonlinear equations (including the complementarity equations) subject to the positiveness of the complementary variables. Hence, the main advantage of this approach is that it avoids the need for discrete decisions and the need for special procedural nonlinear techniques as required by the boundary crossing algorithm. Finally in Chapter 5, we investigated the solving of the resulting complementarity problem using a conventional nonlinear solver and pivotal techniques similar to those proposed by Lemke (1965).

In Chapter 6 we described the globally convergent framework proposed by Wang *et al.* (1996) for solving a constrained system of nonlinear equations by an interior point potential reduction method. Also, we show how the convergence result of this general framework can be applied to the complementarity formulation described in Chapter 5. Based on that, we then applied a modification of the algorithm proposed by Wang and incorporated second order corrections to it in order to solve the complementarity examples used as case studies throughout this research.

7.2 CONTRIBUTIONS

In this section we present a brief summary of what we believe are the main contributions of this work in the various aspects of conditional modeling.

7.2.1 MODELING

- We identify modeling capabilities which support the efficient development of conditional models in both the declarative definition of equation-based models and the procedural execution of methods: *i)* conditional configuration of the model structure, *ii)* conditional compilation, and *iii)* conditional execution of the procedural code of methods.
- We developed consistent syntax and semantics for each of the above modeling capabilities and completed their practical implementation into the ASCEND modeling environment.

In practice, the implementation of conditional modeling tools has proven to be a powerful tool for the purposes of reusability and economy of programming. The application of these tools in many of the current ASCEND modeling libraries provides support for this assessment.

7.2.2 STRUCTURAL ANALYSIS

- We extended Zaher's consistency analysis so that it can be applied to a conditional model without placing restrictions on the number of variables and equations presented in the problem. This algorithm has been implemented within the ASCEND environment.
- We described instances of problems in which the combinatorial complexity of the consistency analysis can be reduced. In particular, the cases of the existence of repeated structures and common incidence pattern among alternatives were discussed:
    1. In our implementation of the consistency analysis, a "pre-analysis" is performed first in order to eliminate those conditional statements whose alternative sets of equations have the same incidence pattern.
    2. The use of a representative incidence matrix significantly reduces the effort to analyze conditional models containing repeated structures. The conditions on which this analysis can be performed have been discussed. However, the implementation of a tool which is able to identify repeated structures and analyze the problem using only a subset of these repeated structures is yet to be completed.

7.2.3 FORMULATION

- We developed a complementarity formulation to represent a conditional model as a complementarity problem. The result of applying our formulation to a disjunctive set of equations is a square system of nonlinear equations. We show that, if the complete set of nonlinear equations (including the complementarity equations) is satisfied, then the solution of the complementarity problem corresponds to a consistent solution to the conditional model.

  The complementarity formulation has shown to be useful in some practical applications. For instance, the thermodynamic library in the ASCEND modeling environment uses a complementarity equation in order to represent the existence or non existence of a phase at some given conditions. This has allowed us to simultaneously find the number of phases and the equilibrium compositions while solving flash calculations with a conventional solver.

7.2.4 SOLUTION

- We provided an efficient implementation of the boundary crossing algorithm as a conditional modeling solution tool. In such an implementation, we have integrated the entities created or used to perform each of the activities required by the algorithm in an object-oriented solving engine: the conditional solver CMSlv. This solver has been incorporated to the ASCEND environment.
- We investigated solving the complementarity problem representing a conditional model using:
  1. A conventional nonlinear solver. We used the ASCEND solver QRSlv which applies a modified Levenberg-Marquardt method.
  2. Pivotal techniques. We studied the use of the Lemke's pivoting rules developed for linear complementarity problems.
  3. Interior point methods. We used the framework developed by Wang and extended his algorithm as well as second order corrections developed by Mehrotra and Gondzio to our complementarity formulation.
- We solved several typical examples in the area of chemical engineering in order to assess the value of the boundary crossing algorithm as well as each of the techniques employed in the solution of the complementarity problems.
- We implemented the interior point methods mentioned above in a computer program called IPSlv. This implementation is also available through the ASCEND solver interface.

7.3 FUTURE WORK

The area of conditional modeling still imposes many challenges to equation-based modeling researchers. The following are a few recommendations for future work on this area:

- In the boundary crossing algorithm implemented in this work, the analysis on a boundary involves the solution to an optimization subproblem. The solution to this subproblem is intended to provide a step which optimally reduces the residuals of all the systems of conditional equations in the neighborhood of the boundary. However, our experience with the solution of conditional models by using pivotal techniques indicates that it may be not necessary to find such an optimal descent direction. In other words, perhaps any step which reduces the residuals of the neighboring systems of equations (any descent as opposed to optimal descent) can give a reasonable movement away from the boundary. Hence, the issue here is to investigate if it is possible to modify the boundary analysis in the boundary crossing algorithm so that we solve a feasibility problem rather than an optimization problem.

- The complementarity problems solved in Chapter 5 and Chapter 6 were formulated by hand using the disjunctive representation of the system of conditional equations. We created conditional modeling tools rich enough to represent the disjunctive equations; what remains to be done is the implementation of a systematic derivation of complementarity problems out of conditional statements representing those disjunctive equations. In that way, we envision a solver which, having available the information provided by conditional statements, could either use the boundary crossing algorithm or generate the complementarity problem and solve this problem by using the various techniques studied in this work.

- A reformulation of our complementarity representation of a conditional model as a mixed complementarity problem (MCP) is desired because it would make suitable the application of a large number of codes and numerical techniques already developed for the solution of MCPs. Future work should be conducted in this direction.

- A conceptual gap still exists between the application of second order interior point methods for the linear programming problem case and their application for the complementarity formulation developed in this work. Future research should be aimed at bridging this gap as far as possible.

- One of the more recent approaches to the discrete event simulation problem was described in the development of the simulation package gPROMS. In that work, Barton (1992) concluded that the analysis of the time dependent behavior of a chemical process requires the system to be decomposed into the continuous-discrete physical behavior of the plant (models) and the external actions imposed on it by its environment (tasks). Because of this decomposition, the physical behavior is specified in a purely declarative language whereas the operation becomes part of the procedural knowledge required to solve the equations. In contrast, we believe that the discrete event simulation problem can be represented by using a purely declarative modeling language.

  The implementation of conditional modeling tools has been the first step towards that goal. However, in order to deal with discrete event simulation problems, it is also necessary to incorporate domain variables, such as time, into the modeling language. The inclusion of domain variables provides a very convenient representation of the time dependent behavior of chemical processes, including any discrete change. The compiling implications of the use of these variables are very interesting. Domain variables can be thought to be defined over an infinite set. ASCEND already supports allowing one to define anything over a finite set. For a finite set, we require the sets to be explicitly defined before compiling commences. Thus, the compiled data structure contains storage space for every variable and equation defined over such a set. On the other hand, for an infinite set the compiler cannot set aside the storage space as the solver will control how many points in time it needs as it solves. What can be compiled, however, is one instance in time of the models. The solver can use the compiled instance to determine the time derivatives and the algebraic variables, given values for the states. Thus, the solver can use this same model repeatedly for different values of the states to integrate forward in time.

APPENDIX A Eligible Set for a Conditional Model Having Alternatives with Different Incident Variables

This appendix presents the formal derivation of the eligible set of variables to be used in the consistency analysis of a conditional model when such a conditional model contains alternatives with different incidence sets and different number of equations among them. By using set theory, we show that the set of variables eligible to become independent variables in the context of the overall conditional problem is given by:

$$e^{k''} = \bigcap_{i}^{s} [E_i^k \cup (M \backslash I_i)]$$

where s is the number of the alternative sets of equations, $e^{k''}$ is the set of eligible variables to be chosen as independent variables for the conditional problem, E_i^k is the eligible set of variables to be chosen as independent variables for the alternative i, M is the maximal set of variables, and I_i is the set of incident variables for the alternative i.

A.1 Notation

Besides the notation already established in Chapter 3, the following definitions are used throughout this demonstration.

A.1.1 Set Operators

comp(A)	Complement	All the elements not in *A*.
$\subset$	Subset	$A \subset B$ means that every element in *A* is also contained in *B*.
$\varnothing$	Empty set	
	Disjoint Sets	$A \cap B = \varnothing$
	Equal Sets	$A \subset B$ and $B \subset A$.

During the derivations, we frequently make use of the Morgan's Theorem:

$$\bigcup_j [comp(A_j)] = comp[\bigcap_j A_j]$$

A.2 Eligible Set in Zaher's Consistency Analysis

In his work, Zaher (1995) considered the necessary conditions for the structural consistency of a conditional model. He show that, for a conditional model in which all the alternatives have the same incident variables, the eligible set of variables for a conditional model is given by:

$$e^k = E_1^k \cap E_2^k \cap \ldots E_s^k = \bigcap_j^s E_j^k \qquad \textbf{(A.1)}$$

A.3 Extension of Zaher’s Consistency Analysis to a General Conditional Model

In a general case, the number of equations in each alternative of a conditional model may change and so may the incidence set of variables. This section presents a derivation of the eligible set of a conditional model for that general case.

A.3.1 Deriving the Eligible Set in a General Conditional Model

In order to derive the eligible set of variables for a general conditional model, we start by finding the eligible set for each alternative in the context of the overall problem through the following analysis.

Assume that an output assignment performed to each individual alternative results in the eligible sets E_i^k for an $i \in \{1 \dots s\}$. Consider the case of alternative number 1. In order to find the “truly” eligible set of variables for alternative 1 in the context of the overall conditional model (all the alternatives) we have to eliminate the elements of E_1^k which are ineligible in any of the alternatives. We will denote the resulting corrected set of eligible variables for alternative 1 as $E_1^k{}'$. We obtain:

$$E_1^k{}' = E_1^k \backslash \{[E_1^k \cap (I_2 \backslash E_2^k)] \cup [E_1^k \cap (I_3 \backslash E_3^k)] \cup \dots [E_1^k \cap (I_S \backslash E_s^k)]\} \qquad \textbf{(A.2)}$$

Here is an explanation of the meaning of (A.2). $(I_2 \backslash E_2^k)$ is the set of variables incident but ineligible to be chosen as independent variables in alternative 2. Hence, $E_1^k \cap (I_2 \backslash E_2^k)$ is the set of eligible variables in alternative 1 which are ineligible in alternative 2. Therefore, $[E_1^k \cap (I_2 \backslash E_2^k)] \cup [E_1^k \cap (I_3 \backslash E_3^k)] \cup \dots [E_1^k \cap (I_S \backslash E_s^k)]$ represents the set of variables which are eligible in alternative 1 but ineligible in some alternative. With that in mind, the corrected set $E_1^k{}'$ is the set of eligible variables for alternative 1 which can be also considered as eligible in the context of the overall conditional model (note the minus operation in (A.2)).

In general, for any alternative $i \in \{1 \dots s\}$ the set of eligible variables for each alternative in the context of the overall conditional problem is given by:

$$E_i^{k\prime} = E_i^k \backslash \left\{ E_i^k \cap \left[\bigcup_{j,\, j \neq i}^{s} (I_j \backslash E_j^k) \right] \right\} \tag{A.3}$$

Finally, the union of these individual sets gives the set of eligible variables for the overall conditional problem:

$$e^{k\prime} = \bigcup_i^s E_i^{k\prime} \tag{A.4}$$

For a general conditional model, equation (A.4) is the equivalent to equation (A.1) for conditional models in which all the alternatives have the same incident variables.

A.3.2 A simplified Approach

We next show that the previous analysis does not have to be performed as described. We demonstrate that, if we augmented the eligible set of each alternative with the nonincident variables of that alternative,

$$E_i^{k\prime\prime} = E_i^k \cup (M \backslash I_i) \tag{A.5}$$

and find the intersection of the augmented sets $E_i^{k\prime\prime}$,

$$e^{k\prime\prime} = \bigcap_i^s E_i^{k\prime\prime} = \bigcap_i^s [E_i^k \cup (M \backslash I_i)] \tag{A.6}$$

then the resulting set $e^{k\prime\prime}$ is completely equivalent to the set $e^{k\prime}$ given by (A.4). Recall that *M* is the maximal set of variables,

$$M = I_1 \cup I_2 \cup \ldots I_s = \bigcup_j^s I_j \tag{A.7}$$

so that $(M \backslash I_i)$ represents the set of nonincidences in alternative *i*.

In oder to demonstrate our assertion, we first carry out the following derivation. From (A.3), we get,

$$E_i^{k\prime} = E_i^k \backslash \left\{ E_i^k \cap \left[\bigcup_{j,\, j \neq i}^{s} (I_j \backslash E_j^k) \right] \right\} \tag{A.8}$$

$$E_i^{k\prime} = E_i^k \backslash \left\{ \bigcup_{j,\, j \neq i}^{s} [E_i^k \cap (I_j \backslash E_j^k)] \right\}$$

$$E_i^{k\prime} = E_i^k \backslash \left\{ \left[\bigcup_{j,\, j \neq i}^{s} E_i^k \cap (I_j \backslash E_j^k) \right] \cup \varnothing \right\}$$

$$E_i^{k\prime} = E_i^k \backslash \left\{ \left[\bigcup_{j,\, j \neq i}^{s} E_i^k \cap (I_j \backslash E_j^k) \right] \cup [E_i^k \cap (I_i \backslash E_i^k)] \right\}$$

$$E_i^{k\prime} = E_i^k \backslash \left\{ \bigcup_{j}^{s} [E_i^k \cap (I_j \backslash E_j^k)] \right\}$$

$$E_i^{k\prime} = E_i^k \backslash \left\{ E_i^k \cap \left[\bigcup_{j}^{s} (I_j \backslash E_j^k) \right] \right\} \tag{A.9}$$

Then we use (A.9) to obtain,

$$E_i^{k\prime} = E_i^k \backslash \left\{ E_i^k \cap \left[\bigcup_{j}^{s} (I_j \backslash E_j^k) \right] \right\} \tag{A.10}$$

$$E_i^{k\prime} = E_i^k \backslash \left\{ E_i^k \cap \left[\bigcup_{j}^{s} [I_j \cup (M \backslash I_j)] \backslash [E_j^k \cup (M \backslash I_j)] \right] \right\}$$

$$E_i^{k\prime} = E_i^k \backslash \left\{ E_i^k \cap \left[\bigcup_{j}^{s} M \backslash [E_j^k \cup (M \backslash I_j)] \right] \right\}$$

$$E_i^{k\prime} = E_i^k \backslash \left\{ E_i^k \cap \left[M \backslash \left(\bigcap_j^s E_j^k \cup (M \backslash I_j) \right) \right] \right\}$$

$$E_i^{k\prime} = E_i^k \backslash \left\{ E_i^k \cap \left[M \backslash \left(\bigcap_j^s E_j^k \cup (M \backslash I_j) \right) \right] \right\}$$

$$E_i^{k\prime} = E_i^k \backslash \left\{ [E_i^k \cap M] \backslash \left[E_i^k \cap \left(\bigcap_j^s E_j^k \cup (M \backslash I_j) \right) \right] \right\}$$

$$E_i^{k\prime} = E_i^k \cap \left[\bigcap_j^s E_j^k \cup (M \backslash I_j) \right] \qquad \textbf{(A.11)}$$

Thus, by substituting (A.6) in (A.11):

$$E_i^{k\prime} = E_i^k \cap e^{k\prime\prime} \qquad \textbf{(A.12)}$$

and from (A.4):

$$e^{k\prime} = \bigcup_i^s [E_i^k \cap e^{k\prime\prime}] \qquad \textbf{(A.13)}$$

which is equivalent to:

$$e^{k\prime} = \left(\bigcup_i^s E_i^k \right) \cap e^{k\prime\prime} \qquad \textbf{(A.14)}$$

Finally, since $e^{k\prime\prime}$ is always a subset of $\left(\bigcup_i^s E_i^k \right)$, we obtain,

$$e^{k\prime} = e^{k\prime\prime} \qquad \textbf{(A.15)}$$

Q.E.D

(A.15) means that the derivation of the set of eligible variables for a conditional problem presented in A.3.1 can be substituted by the combined use of (A.5) and (A.6).

A.4 Proof of the Reduction to the Particular Case

We finish our demonstration by showing that both (A.4) and (A.6), reduces to (A.1) when the incidences are the same for all the alternatives.

A.4.1 Reduction from (A.4)

Start with equation (A.3)

$$E_i^{k\prime} = E_i^k \backslash \left\{ E_i^k \cap \left[\bigcup_{j,\, j \neq i}^{s} (I_j \backslash E_j^k) \right] \right\}$$

since

$$I_1 = I_2 = \dots = I_s = I = M \qquad \textbf{(A.16)}$$

then

$$(I_j \backslash E_j^k) = (I \backslash E_j^k) = (M \backslash E_j^k) = comp(E_j^k) \qquad \textbf{(A.17)}$$

By using Morgan’s Theorem:

$$\bigcup_j [comp(E_j^k)] = comp\left[\bigcap_j E_j^k \right] \qquad \textbf{(A.18)}$$

we get the following equivalence:

$$\bigcup_{j,\, j \neq i}^{s} (I_j \backslash E_j^k) = I \backslash \bigcap_{j,\, j \neq i}^{s} (E_j^k) \qquad \textbf{(A.19)}$$

Substituting (A.19) in (A.3):

$$E_i^{k\prime} = E_i^k \backslash \left\{ E_i^k \cap \left[I \backslash \bigcap_{j,\, j \neq i}^{s} (E_j^k) \right] \right\} \qquad \textbf{(A.20)}$$

and further simplifying:

$$E_i^{k\prime} = E_i^k \backslash \left\{ [E_i^k \cap I] \backslash \left[E_i^k \cap \left(\bigcap_{j,\, j \neq i}^{s} (E_j^k) \right) \right] \right\} \quad \textbf{(A.21)}$$

$$E_i^{k\prime} = E_i^k \backslash \left\{ E_i^k \backslash \left[\bigcap_{i}^{s} (E_j^k) \right] \right\} \quad \textbf{(A.22)}$$

$$E_i^{k\prime} = \bigcap_{i}^{s} (E_j^k) \quad \textbf{(A.23)}$$

From (A.1):

$$E_i^{k\prime} = e^k \quad \textbf{(A.24)}$$

Finally, from (A.4):

$$e^{k\prime} = \bigcup_{i}^{s} E_i^{k\prime} = \bigcup_{i}^{s} e^k = e^k \quad \textbf{(A.25)}$$

Q.E.D.

A.4.2 Reduction from (A.6)

Starting with (A.5)

$$E_i^{k\prime\prime} = E_i^k \cup (M \backslash I_i) \quad \textbf{(A.26)}$$

and since

$$I_1 = I_2 = \ldots = I_s = I = M \quad \textbf{(A.27)}$$

then the set of nonincidences is null for all the alternatives:

$$M \backslash I_i = \varnothing \quad \textbf{(A.28)}$$

Hence,

$$E_i^{k\prime\prime} = E_i^k \quad \textbf{(A.29)}$$

and, finally,

$$e^k = \bigcap_i^s E_i^k = \bigcap_i^s E_i^{k}{}'' = e^{k}{}'' \qquad \textbf{(A.30)}$$

Q.E.D.

APPENDIX B Derivation of the Optimization Subproblem of the Boundary Crossing Algorithm

In this appendix, we present the derivation of Equation (4.7) from Equation (4.6). Equation (4.7) represents the optimization subproblem that we need to solve when the boundary crossing algorithm encounters a boundary in the solution path.

B.1 The Derivation

Zaher (1991,1995) describes the derivation of Equation (4.7) as follows. Let us start from Equation (4.6):

$$\min \begin{pmatrix} \max & \nabla_x \phi_i(\underline{a})^T \cdot \underline{d} \\ s.t. & i \in B \end{pmatrix}$$
$$s.t. \quad \underline{d}^T \cdot \underline{d} \le 1$$

where B is the set of all of the subregions in the neighborhood of a boundary, $\nabla_x \phi_i(\underline{a})$ is the gradient vector of the objective function for the neighboring subregion i, and $\underline{d}$ is the descent direction which constitutes the solution to the problem (4.6). Let us now define the matrix N whose columns are the vectors of gradients of the subregions in the neighborhood of the current boundary. That is, column i of N is the vector $\nabla_x \phi_i(\underline{a}), \forall i \in B$. Hence, the inner maximization problem of (4.6) can be reformulated as finding the value of a scalar β such that:

$$\beta \underline{e} \ge N^T \underline{d} \tag{B.1}$$

where $\underline{e}$ is a vector with all its elements equal to one. Then, Equation (4.6) can be rewritten as:

$$\begin{aligned} \min \quad & \beta \\ s.t. \quad & \beta \underline{e} \ge N^T \underline{d} \\ & \underline{d}^T \cdot \underline{d} \le 1 \end{aligned} \tag{B.2}$$

The real-valued Lagrange function, L, of problem (B.2) is given by:

$$L(\beta, \underline{d}, \underline{\alpha}, \pi) = \beta - \underline{\alpha}^T(\beta \underline{e} - N^T d) - \pi(1 - d^T d) \tag{B.3}$$

where $\underline{\alpha}$ is the vector of multipliers corresponding to first of the inequality constraints in (B.2) and π is the multiplier (scalar) corresponding to the second inequality of (B.2). Hence, the dual of the problem (B.2) is given by:

$$\begin{aligned} max \quad & \beta - \underline{\alpha}^T(\beta \underline{e} - N^T d) - \pi(1 - d^T d) \\ s.t. \quad & \underline{e}^T \cdot \underline{\alpha} = 1 \\ & 2\pi \underline{d} = -N\underline{\alpha} \\ & \pi \geq 0,\ \alpha_i \geq 0,\ \forall i \in B \end{aligned} \qquad \textbf{(B.4)}$$

The equality constraints of (B.4) were obtained from the first order conditions of the Lagrangian function with respect to the variables β and $\underline{d}$. After further simplification, (B.4) reduces to:

$$\begin{aligned} max \quad & -\pi(\underline{d}^T \cdot \underline{d} + 1) \\ & 2\pi \underline{d} = -N \cdot \underline{\alpha} \\ & \underline{e}^T \cdot \underline{\alpha} = 1 \\ & \pi \geq 0,\ \alpha_i \geq 0,\ \forall i \in B \end{aligned} \qquad \textbf{(B.5)}$$

which is equivalent to:

$$\begin{aligned} min \quad & \pi(\underline{d}^T \cdot \underline{d} + 1) \\ & 2\pi \underline{d} = -N \cdot \underline{\alpha} \\ & \underline{e}^T \cdot \underline{\alpha} = 1 \\ & \pi \geq 0,\ \alpha_i \geq 0,\ \forall i \in B \end{aligned} \qquad \textbf{(B.6)}$$

Finally, it should be recognized that, as long as the problem (B.6) is bounded and the solution is non trivial ($\pi > 0$), the multiplier π will have no effect in the solution vector $\underline{d}$ of the optimization problem (B.6). Therefore, (B.6) reduces to equation (4.7):

$$\begin{aligned} min \quad & \underline{d}^T \cdot \underline{d} \\ s.t. \quad & \underline{d} = -N \cdot \underline{\alpha} \\ & \underline{e}^T \cdot \underline{\alpha} = 1 \\ & \alpha_i \geq 0,\ \forall i \in B \end{aligned}$$

APPENDIX C ASCEND Models of the Examples Solved with the Boundary Crossing Algorithm

In this appendix, we show a representative section of the ASCEND models for the examples described in Chapter 4.

C.1 The Models

Example 1 Fluid Transition (Zaher,1995).

```
CONDITION
      bound: (Pd - Pf) * 1.0 {atm^-1} < Mf - 1.0;
END CONDITION;

sonic_flow <==> SATISFIED(bound,1e-08);

subsonic: Pf = Pd;
sonic: Mf = 1.0;

WHEN (sonic_flow)
      CASE TRUE:
            USE sonic;
      CASE FALSE:
            USE subsonic;
END WHEN;
```

Example 2 Phase Equilibria (Zaher,1995).

```
components :== [‘B’,’E’,’W’];
phases :== [‘A’,’O’,’V’];

CONDITION
      FOR i IN phases CREATE
            bound[i]: SUM[y[i][j] | j IN components ]+
                                                phi[i] >= 1.0;
      END FOR;
END CONDITION;

FOR i IN phases CREATE
      sum[i]: SUM[y[i][j] | j IN components ] = 1.0;
      frac[i]: phi[i] = 0.0;
      exist[i] <==> SATISFIED(bound[i],1e-08);

      WHEN (exist[i])
            CASE TRUE:
                  USE sum[i];
            CASE FALSE:
                  USE frac[i];
      END WHEN;
END FOR;
```

Example 3 Heat Exchanger (Zaher,1995)

```
components :== [‘B’,’P5’,’H’];

CONDITION
      bound1: SUM[x[0][i] | i IN components] + phi[0] >= 1.0;
      bound2: SUM[x[2][i] | i IN components] + phi[1] >= 1.0;
END CONDITION;

liq_exist[1] <==> SATISFIED(bound1,1e-08);
liq_exist[2] <==> SATISFIED(bound2,1e-08);

sum0: SUM[x[0][i] | i IN components] = 1.0;
frac0: phi[0] = 0.0;

sum2: SUM[x[2][i] | i IN components] = 1.0;
frac1: phi[1] = 0.0;

p1: eta[1] = 0.5;
p2: eta[2] = 0.5;

sum1: SUM[x[1][i] | i IN components] = 1.0;
p12: eta[1] = 0.0;

WHEN (liq_exist[1])
      CASE TRUE:
            USE sum0;
            USE sum1;
      CASE FALSE:
            USE frac0;
            USE p12;
END WHEN;

WHEN (liq_exist[2])
      CASE TRUE:
            USE sum2;
            USE p1;
      CASE FALSE:
            USE frac1;
            USE p2;
END WHEN;
```

Example 4 Pipeline Network (Bullard and Biegler, 1992). Model of a pipe with no valve.

```
      positive_flow     IS_A  boolean_var;

      CONDITION
            bound: Q >= 0.0 {gpm};
      END CONDITION;

      positive_flow <==> SATISFIED(bound,1e-08 {gpm});

      positive_head: H = k * sqr(Q);
      negative_head: H = -k * sqr(Q);

      WHEN (positive_flow)
            CASE TRUE:
                  USE positive_head;
            CASE FALSE:
                  USE negative_head;
      END WHEN;
```

Example 5 Simple L-V Flash Calculation (King, 1980).

```
(* Rachford-Rice *)

      SUM[ ( z[i] * ( K[i] - 1 ) ) / ( ((K[i]-1)* V_F) + 1 )
                                  | i IN components ] = R - V_F;

(* Conditional Equations *)
      CONDITION
            bound1: R <= 0.0;
            bound2: R <= 1.0;
      END CONDITION;

(* Logical Relations *)
      bol1 <==> SATISFIED(bound1,1e-08);
      bol2 <==> SATISFIED(bound2,1e-08);

(* Variant Equations*)
      liq_eq: V_F = 0.0;
      two_eq: V_F = R;
      vap_eq: V_F = 1.0;
```

```
(* Disjunctions *)
      WHEN (bol1,bol2)
            CASE TRUE,TRUE:
                  USE liq_eq;
            CASE FALSE,TRUE:
                  USE two_eq;
            CASE FALSE,FALSE:
                  USE vap_eq;
            OTHERWISE:
      END WHEN;
```

Example 6 Linear Mass Balance (Grossmann and Turkay, 1996).

```
      CONDITION
            bound1: Fmain <= B[1];
            bound2: Fmain >= B[2];
      END CONDITION;

      bol1 <==> SATISFIED(bound1,1e-08{lb_mole/hour});
      bol2 <==> SATISFIED(bound2,1e-08{lb_mole/hour});

(* Variant Equations *)

      eq1a: Fsub1 = a[1] * Fmain;
      eq1b: Fsub2 = b[1] * Fmain;

      eq2a: Fsub1 = a[2] * Fmain;
      eq2b: Fsub2 = b[2] * Fmain;

      eq3a: Fsub1 = a[3] * Fmain;
      eq3b: Fsub2 = b[3] * Fmain;

(* Disjunction  *)

      WHEN (bol1,bol2)
            CASE TRUE,FALSE:
                  USE eq1a;
                  USE eq1b;
            CASE FALSE,FALSE:
                  USE eq2a;
                  USE eq2b;
            CASE FALSE,TRUE:
                  USE eq3a;
                  USE eq3b;
      END WHEN;
```

APPENDIX D Complementarity Equations of the Examples of Chapter 5

This appendix presents the complementarity equations (or a representative part of them) for each of the examples solved in Chapter 5.

D.1 Complementarity Equations for the Examples Solved in Chapter 5

Example 1 Fluid Transition (Zaher,1995).

A single complementarity equation is required. The definition of the positive residuals and the complementarity equation are given by:

$$\begin{gathered} 1 - M_f = p_{1_1} \\ p_f - p_d = p_{2_1} \\ \begin{bmatrix} p_{1_1} = 0 \\ p_{2_1} \geq 0 \end{bmatrix} \vee \begin{bmatrix} p_{2_1} = 0 \\ p_{1_1} \geq 0 \end{bmatrix} \Rightarrow \begin{matrix} p_{1_1} \cdot p_{2_1} = 0 \\ p_{1_1}, p_{2_1} \geq 0 \end{matrix} \end{gathered} \qquad \textbf{(D.1)}$$

Since the problem contains only one condition in each disjunctive term, our complementarity representation reduces to the standard complementarity formulation.

Example 2 Phase Equilibria (Zaher,1995).

The representation of the existence-non existence of each phase is given by the following set of equations:

$$\begin{gathered} \phi_A = p_{1_1} \\ 1 - \sum_{i \in C} y_{i_A} = p_{2_1} \\ \begin{bmatrix} p_{1_1} = 0 \\ p_{2_1} \geq 0 \end{bmatrix} \vee \begin{bmatrix} p_{2_1} = 0 \\ p_{1_1} \geq 0 \end{bmatrix} \Rightarrow \begin{matrix} p_{1_1} \cdot p_{2_1} = 0 \\ p_{1_1}, p_{2_1} \geq 0 \end{matrix} \end{gathered} \qquad \textbf{(D.2)}$$

There are three possible phases, but there is only one equation in each disjunction, and, therefore, only one complementarity equation for each of phases is required. Each of the complementarity equations is similar to the complementarity equation of Example 1.

Example 3 Heat Exchanger (Zaher,1995)

In order to obtain the complementarity formulation of this example, we start by recognizing that we can decouple the disjunctive statement of Example 4.3 in two independent disjunctions:

$$\begin{bmatrix} \phi_0 = 0 \\ \eta_1 = 0 \\ \sum_{i \in C} x_{i_0} + \phi_0 < 1 \end{bmatrix} \vee \begin{bmatrix} \sum_{i \in C} x_{i_0} = 1 \\ \sum_{i \in C} x_{i_1} = 1 \\ \sum_{i \in C} x_{i_0} + \phi_0 \geq 1 \end{bmatrix} \tag{D.3}$$

and

$$\begin{bmatrix} \phi_1 = 0 \\ \eta_2 = 0.5 \\ \sum_{i \in C} x_{i_2} + \phi_1 < 1 \end{bmatrix} \vee \begin{bmatrix} \sum_{i \in C} x_{i_2} = 1 \\ \eta_1 = 0.5 \\ \sum_{i \in C} x_{i_2} + \phi_1 \geq 1 \end{bmatrix} \tag{D.4}$$

Hence, the complementarity equations for these disjunctions are:

$$\begin{gathered} \phi_0 = p_{1_{11}} \qquad 1 - \sum_{i \in C} x_{i_0} = p_{2_{11}} \\ \eta_1 = p_{1_{21}} \qquad 1 - \sum_{i \in C} x_{i_1} = p_{2_{21}} \\ \begin{bmatrix} p_{1_{11}} = 0 \\ p_{1_{21}} = 0 \end{bmatrix} \vee \begin{bmatrix} p_{2_{11}} = 0 \\ p_{2_{21}} = 0 \end{bmatrix} \Rightarrow \begin{matrix} p_{1_{11}} \cdot p_{2_{11}} + p_{1_{21}} \cdot p_{2_{21}} = 0 \\ p_{1_{11}} \cdot p_{2_{21}} + p_{1_{21}} \cdot p_{2_{11}} = 0 \end{matrix} \\ p_{i_{j1}} \geq 0 \qquad \forall i \in [1\ldots2], j \in [1\ldots2] \end{gathered} \tag{D.5}$$

and

$$\phi_1 = p_{1_{12}} \qquad 1 - \sum_{i \in C} x_{i_2} = p_{2_{12}}$$
$$\eta_2 - 0.5 = p_{1_{22}} \qquad 0.5 - \eta_1 = p_{2_{22}}$$
$$\begin{bmatrix} p_{1_{12}} = 0 \\ p_{1_{22}} = 0 \end{bmatrix} \vee \begin{bmatrix} p_{2_{12}} = 0 \\ p_{2_{22}} = 0 \end{bmatrix} \Rightarrow \begin{matrix} p_{1_{12}} \cdot p_{2_{12}} + p_{1_{22}} \cdot p_{2_{22}} = 0 \\ p_{1_{12}} \cdot p_{2_{22}} + p_{1_{22}} \cdot p_{2_{12}} = 0 \end{matrix} \tag{D.6}$$
$$p_{i_{j2}} \geq 0 \qquad \forall i \in [1\ldots2], j \in [1\ldots2]$$

Example 4 Pipeline Network (Bullard and Biegler, 1992).

1) Arcs with no valve

$$H_{ij} + K \cdot Q_{ij}^{\ 2} = p_{1_1}$$
$$K \cdot Q_{ij}^{\ 2} - H_{ij} = p_{2_1}$$
$$Q_{ij} = p_{1_2} - p_{2_2}$$
$$\begin{bmatrix} p_{1_1} = 0 \\ p_{1_2} = 0 \end{bmatrix} \vee \begin{bmatrix} p_{2_1} = 0 \\ p_{2_2} = 0 \end{bmatrix} \Rightarrow \begin{matrix} p_{1_1} \cdot p_{2_1} + p_{1_2} \cdot p_{2_2} = 0 \\ p_{1_1} \cdot p_{2_2} + p_{1_2} \cdot p_{2_1} = 0 \end{matrix} \tag{D.7}$$
$$p_{k_q} \geq 0 \qquad \forall k \in [1\ldots2], q \in [1\ldots2]$$

2)Arcs with check valve

$$K \cdot Q^2{}_{ij} = p_{1_1}$$
$$K \cdot Q^2{}_{ij} - H_{ij} = p_{2_1}$$
$$H_{ij} = p_{1_2} - p_{2_2}$$
$$\begin{bmatrix} p_{1_1} = 0 \\ p_{1_2} = 0 \end{bmatrix} \vee \begin{bmatrix} p_{2_1} = 0 \\ p_{2_2} = 0 \end{bmatrix} \Rightarrow \begin{matrix} p_{1_1} \cdot p_{2_1} + p_{1_2} \cdot p_{2_2} = 0 \\ p_{1_1} \cdot p_{2_2} + p_{1_2} \cdot p_{2_1} = 0 \end{matrix} \tag{D.8}$$
$$p_{k_q} \geq 0 \qquad \forall k \in [1\ldots2], q \in [1\ldots2]$$

Example 5 Simple L-V Flash Calculation (King, 1980).

The behavior described in Example 4.5 can be represented in terms of the following disjunctive statement:

$$\sum_i \frac{z_i \cdot (K_i - 1)}{(K_i - 1) \cdot (V/F) + 1} = R - V/F$$
$$\begin{bmatrix} V/F = 0 \\ R \le 0 \end{bmatrix} \vee \begin{bmatrix} V/F = R \\ 0 \le R \le 1 \end{bmatrix} \vee \begin{bmatrix} V/F = 1 \\ R \ge 1 \end{bmatrix} \qquad \textbf{(D.9)}$$

This example includes a disjunction with three disjunctive terms in it, but a complementarity formulation can still be obtained. Also, in this case, some of the residual variables *p* will appear in more than one disjunctive term and a representation including an index *i* for each disjunctive term could be confusing. For that reason, we use the variables *p* indexed in successive order as follows:

$$V/F = p_1 \qquad V/F = R + p_2 - p_3 \qquad V/F = 1 - p_4$$
$$R = p_5 - p_6 \qquad R = 1 + p_7 - p_8 \qquad \textbf{(D.10)}$$

The disjunctive representation in terms of the residuals is:

$$\begin{bmatrix} p_1 = 0 \\ p_5 = 0 \\ p_3 = 0 \\ p_7 = 0 \end{bmatrix} \vee \begin{bmatrix} p_2 = 0 \\ p_6 = 0 \\ p_3 = 0 \\ p_7 = 0 \end{bmatrix} \vee \begin{bmatrix} p_2 = 0 \\ p_6 = 0 \\ p_4 = 0 \\ p_8 = 0 \end{bmatrix} \qquad \textbf{(D.11)}$$
$$p_j \ge 0 \qquad \forall j \in [1\ldots 8]$$

Note that several conditions appear in two different disjunctive terms. We could generate the complementarity equations as we described in section 5.2.1.3. The formulation obtained would still be consistent. However, we took advantage of the structure of the disjunctive representation given above in order to simplify the formulation. The following structure can be identified:

$$\begin{bmatrix} A \\ B \end{bmatrix} \vee \begin{bmatrix} C \\ B \end{bmatrix} \vee \begin{bmatrix} C \\ D \end{bmatrix} \Rightarrow (A \wedge B) \vee (C \wedge B) \vee (C \wedge D) \tag{D.12}$$

and converting from Disjunctive Normal Form to Conjunctive Normal Form:

$$(A \wedge B) \vee (C \wedge B) \vee (C \wedge D) \Rightarrow (A \vee C) \wedge (B \vee C) \wedge (B \vee D) \tag{D.13}$$

This derivation tells us that, in order to represent the three-term disjunctive term, we can use the union of three two-term disjunctions:

$$\left\{ \begin{bmatrix} p_1 = 0 \\ p_5 = 0 \end{bmatrix} \vee \begin{bmatrix} p_2 = 0 \\ p_6 = 0 \end{bmatrix} \right\} \wedge \left\{ \begin{bmatrix} p_3 = 0 \\ p_7 = 0 \end{bmatrix} \vee \begin{bmatrix} p_2 = 0 \\ p_6 = 0 \end{bmatrix} \right\} \wedge \left\{ \begin{bmatrix} p_3 = 0 \\ p_7 = 0 \end{bmatrix} \vee \begin{bmatrix} p_4 = 0 \\ p_8 = 0 \end{bmatrix} \right\} \tag{D.14}$$

$$p_j \geq 0 \qquad \forall j \in [1 \dots 8]$$

which is equivalent to

$$\left\{ \begin{matrix} p_1 \cdot p_2 + p_5 \cdot p_6 = 0 \\ p_1 \cdot p_6 + p_5 \cdot p_2 = 0 \end{matrix} \right\} \wedge \left\{ \begin{matrix} p_3 \cdot p_2 + p_7 \cdot p_6 = 0 \\ p_3 \cdot p_6 + p_7 \cdot p_2 = 0 \end{matrix} \right\} \wedge \left\{ \begin{matrix} p_3 \cdot p_4 + p_7 \cdot p_8 = 0 \\ p_3 \cdot p_8 + p_7 \cdot p_4 = 0 \end{matrix} \right\} \tag{D.15}$$

$$p_j \geq 0 \qquad \forall j \in [1 \dots 8]$$

If we apply that simplification for this example, the number of terms in our complementarity representation decreases significantly: strictly, a disjunction with three terms and four conditions in each term requires $4^3 = 64$ trilinear terms for its representation. On the other hand, the previous derivation tells us that 12 bilinear terms are enough for the representation of this particular problem.

We still need to find out how to distribute the 12 bilinear terms through the 4 complementarity equations required to obtain a square system of equations. Such a distribution is not unique, and we can use any set of complementarity equations which does not introduce numerical singularities (any set in which it can be proved that there is a possible pivot for each complementarity equation in the Jacobian matrix) to the system. The distribution can be as simple as the following:

$$\begin{aligned} p_1 \cdot p_2 + p_5 \cdot p_6 &= 0 \\ p_1 \cdot p_6 + p_5 \cdot p_2 &= 0 \\ p_3 \cdot p_2 + p_7 \cdot p_6 + p_3 \cdot p_4 + p_7 \cdot p_8 &= 0 \\ p_3 \cdot p_6 + p_7 \cdot p_2 + p_3 \cdot p_8 + p_7 \cdot p_4 &= 0 \\ p_j \geq 0 \qquad \forall j &\in [1 \ldots 8] \end{aligned} \tag{D.16}$$

in which the terms of two disjunctions are joined to generate two of the equations, or a more thoughtful one like:

$$\begin{aligned} p_1 \cdot p_2 + p_3 \cdot p_6 + p_7 \cdot p_8 &= 0 \\ p_1 \cdot p_6 + p_3 \cdot p_2 + p_7 \cdot p_4 &= 0 \\ p_5 \cdot p_2 + p_7 \cdot p_6 + p_3 \cdot p_4 &= 0 \\ p_5 \cdot p_6 + p_7 \cdot p_2 + p_3 \cdot p_8 &= 0 \\ p_j \geq 0 \qquad \forall j &\in [1 \ldots 8] \end{aligned} \tag{D.17}$$

in which the complementarity pairs are arranged to avoid repeated indices in the complementarity equations. When incorporated with the rest of the system, both sets of complementarity equations provide the correct solution to the problem. The number of iterations that we reported in Chapter 5 corresponds to the straightforward formulation given first.

Example 6 Linear Mass Balance (Grossmann and Turkay, 1996).

Even though the original disjunctive problem is linear, the introduction of the complementarity equations leaves us with a nonlinear system of equations. Also, as in Example 4.5, this example contains disjunctions with 3 disjunctive terms. As a consequence, the formulation once again becomes complicated. Units 1 through 5 (those containing two equations in each term) can be represented by similar sets of disjunctive equations. Here we only show the complementarity equations for unit operation 1. The equations for unit operation 2 to 5 are very similar to the equations generated for this one. For the case of unit 6 (which has only one equation in each disjunctive term), the disjunctive statement and the complementarity equations are similar from those obtained for the example of the simple flash calculation.

The simplification process for this example is also the same as that given in Example 5.

Definition of the residual and slack variables:

$$\begin{array}{lll} F_6 = 1.1 \cdot F_7 + p_1 & F_6 = 1.5 \cdot F_7 + p_2 - p_3 & F_6 = 1.2 \cdot F_7 - p_4 \\ F_{10} = 0.05 \cdot F_7 + p_5 & F_{10} = 0.1 \cdot F_7 + p_6 - p_7 & F_{10} = 0.2 \cdot F_7 - p_8 \\ & F_7 = 50 + p_9 - p_{10} \qquad F_7 = 80 + p_{11} - p_{12} & \end{array} \tag{D.18}$$

Disjunctive statement in terms of the residual and slack variables:

$$\begin{bmatrix} p_1 = 0 \\ p_5 = 0 \\ p_9 = 0 \\ p_2 = 0 \\ p_6 = 0 \\ p_{11} = 0 \end{bmatrix} \vee \begin{bmatrix} p_3 = 0 \\ p_7 = 0 \\ p_{10} = 0 \\ p_2 = 0 \\ p_6 = 0 \\ p_{11} = 0 \end{bmatrix} \vee \begin{bmatrix} p_3 = 0 \\ p_7 = 0 \\ p_{10} = 0 \\ p_4 = 0 \\ p_8 = 0 \\ p_{12} = 0 \end{bmatrix} \tag{D.19}$$

$$p_j \ge 0 \qquad \forall j \in [1 \dots 12]$$

After converting to Conjunctive Normal Form:

$$\left\{\begin{bmatrix} p_1 = 0 \\ p_5 = 0 \\ p_9 = 0 \end{bmatrix} \vee \begin{bmatrix} p_3 = 0 \\ p_7 = 0 \\ p_{10} = 0 \end{bmatrix}\right\} \wedge \left\{\begin{bmatrix} p_2 = 0 \\ p_6 = 0 \\ p_{11} = 0 \end{bmatrix} \vee \begin{bmatrix} p_3 = 0 \\ p_7 = 0 \\ p_{10} = 0 \end{bmatrix}\right\} \wedge \left\{\begin{bmatrix} p_2 = 0 \\ p_6 = 0 \\ p_{11} = 0 \end{bmatrix} \vee \begin{bmatrix} p_4 = 0 \\ p_8 = 0 \\ p_{12} = 0 \end{bmatrix}\right\} \qquad \textbf{(D.20)}$$

$$p_j \geq 0 \qquad \forall j \in [1 \ldots 12]$$

One possible set of complementarity equations:

$$\begin{gathered}
p_1 \cdot p_3 + p_2 \cdot p_4 + p_9 \cdot p_7 + p_6 \cdot p_{10} + p_{11} \cdot p_{12} = 0 \\
p_1 \cdot p_7 + p_2 \cdot p_8 + p_9 \cdot p_{10} + p_6 \cdot p_3 + p_{11} \cdot p_4 = 0 \\
p_1 \cdot p_{10} + p_2 \cdot p_{12} + p_9 \cdot p_3 + p_6 \cdot p_7 + p_{11} \cdot p_8 = 0 \\
p_5 \cdot p_3 + p_6 \cdot p_4 + p_{11} \cdot p_{10} + p_2 \cdot p_7 = 0 \\
p_5 \cdot p_7 + p_6 \cdot p_8 + p_{11} \cdot p_3 + p_2 \cdot p_{10} = 0 \\
p_5 \cdot p_{10} + p_6 \cdot p_{12} + p_{11} \cdot p_7 + p_2 \cdot p_3 = 0 \\
p_j \geq 0 \qquad \forall j \in [1 \ldots 12]
\end{gathered} \qquad \textbf{(D.21)}$$

The number of iterations reported in Chapter 5 corresponds to the solution of the system of equations including the set of complementarity equations (C.21), but some other alternative sets were also successfully used to obtain the solution to the problem

APPENDIX E Proof of the descent of the potential function in Wang's Algorithm

This appendix presents the demonstration developed by Wang *et al.* (1996). Based on the work of Kojima *et al.* (1994), Wang shows that perturbed Newton direction d is a descent direction of the function potential function $\psi(\omega)$ such that:

$$\psi(\omega + \lambda d) - \psi(\omega) \leq -\alpha\lambda(1 - \sigma)(\zeta - n_2)$$

where $\alpha \in (0,1)$, λ is a scalar and there exists a $\bar{\lambda}>0$ such that, for all $\lambda \in (0,\bar{\lambda})$, we get $\omega + \lambda d \in \Omega_\Sigma$.

E.1 Demonstration

Wang *et al*. (1996) establish some important properties about the perturbed Newton direction calculated within the potential reduction algorithm. These properties are established in Lemma 1 below.

Lemma 1. Let $\zeta > n_2$ be given. Under the assumptions

- $\Omega_\Sigma \neq 0$ and $\Omega_\Sigma = \{\omega \in \Omega | G(\omega) > 0\}$ and
- H is continuously differentiable on the set Ω_Σ,

the following properties are valid for an arbitrary vector $\omega \in \Omega_\Sigma$ and any scalar $\alpha \in (0, 1)$:

(i) $\Psi(\omega) \geq (\zeta - n_2) \cdot \log(\nu(\omega)) + n_2 \log(n_2)$

(ii) ψ is continuously differentiable at ω and has a gradient:

$$\nabla\psi(\omega) = \frac{\zeta}{\nu(\omega)} \cdot (2\nabla F(\omega)F(\omega) + \nabla G(\omega)e) - \nabla G(\omega)G(\omega)^{-1}$$

(iii) If d is a solution of

$$H'(\omega)d = \begin{bmatrix} F'(\omega)d \\ G'(\omega)d \end{bmatrix} = \begin{bmatrix} -F(\omega) \\ -G(\omega) + \sigma\mu(\omega)e \end{bmatrix}$$

then

$$\nabla\psi(\omega)^T d \leq -(1 - \sigma) + (\zeta - n_2)\,;$$

hence there exists a scalar $\bar{\lambda}$>0 such that for all $\lambda \in (0,\bar{\lambda})$

$$\omega + \lambda d \in \Omega_\Sigma,$$

$$\psi(\omega + \lambda d) - \psi(\omega) \leq -\alpha\lambda(1 - \sigma)(\zeta - n_2) < 0$$

Proof. The inequality *(i)* is frequently used in the study of interior point methods. This proof is taken from Kojima *et al.* (1994):

Starting with the definitions of the potential function $\psi(\omega)$:

$$\psi(\omega) = \zeta \cdot \log(F(\omega)^T F(\omega) + e^T G(\omega)) - \sum_i^{n_2} \log G_i(\omega)$$

and the measure of the error $\nu(\omega)$:

$$\nu(\omega) = F(\omega)^T F(\omega) + e^T G(\omega)$$

Then, we get

$$\psi(\omega) = \zeta \cdot \log(\nu(\omega)) - \sum_i^{n_2} \log G_i(\omega) \quad \textbf{(E.1)}$$

$$\psi(\omega) = (\zeta - n_2) \cdot \log(\nu(\omega)) + n_2 \cdot \log(\nu(\omega)) - \sum_i^{n_2} \log G_i(\omega) \quad \textbf{(E.2)}$$

$$\psi(\omega) \geq (\zeta - n_2) \cdot \log(\nu(\omega)) + n_2 \cdot \log(e^T G(\omega)) - \sum_i^{n_2} \log G_i(\omega) \quad \textbf{(E.3)}$$

$$\psi(\omega) \geq (\zeta - n_2) \cdot \log(\nu(\omega)) + n_2 \cdot \log\left(\frac{(e^T G(\omega))/n_2}{\left(\prod_{i=1}^{n_2} G_i(\omega)\right)^{1/n_2}}\right) + n_2 \log(n_2) \quad \textbf{(E.4)}$$

$$\Psi(\omega) \geq (\zeta - n_2) \cdot \log(\nu(\omega)) + n_2 \log(n_2) \quad \textbf{(E.5)}$$

Here the last inequality follows from

$$n_2 \cdot \log\left(\frac{(e^T G(\omega))/n_2}{\left(\prod_{i=1}^{n_2} G_i(\omega)\right)^{1/n_2}}\right) \geq 0 \quad \textbf{(E.6)}$$

The identity in (ii) follows from the following computation. Consider the definition of

$\psi(\omega)$ given in (E.1). For each $j = 1,2,...n_1+n_2$,

$$\frac{\partial\psi(\omega)}{\partial\omega_j} = \frac{\zeta}{\nu(\omega)} \cdot \left(\sum_{i=1}^{n_1} 2F_i(\omega) \cdot \frac{\partial F_i(\omega)}{\partial\omega_j} + \sum_{i=1}^{n_2} \frac{\partial G_i(\omega)}{\partial\omega_j}\right) - \sum_{i=1}^{n_2} G_i(\omega)^{-1} \cdot \frac{\partial G_i(\omega)}{\partial\omega_j} \qquad \textbf{(E.7)}$$

and, therefore,

$$\nabla\psi(\omega) = \frac{\zeta}{\nu(\omega)} \cdot (2\nabla F(\omega)F(\omega) + \nabla G(\omega)e) - \nabla G(\omega)G(\omega)^{-1} \qquad \textbf{(E.8)}$$

To prove the first inequality in *(iii)*, we use *(ii)* and the definition of $\mu(\omega) = e^T G(\omega)/n_2$ to get

$$\nabla\psi(\omega)^T d = \frac{\zeta}{\nu(\omega)} \cdot (2F(\omega)^T F'(\omega)d + e^T G'(\omega)d) - (G(\omega)^{-1})^T G'(\omega)d \qquad \textbf{(E.9)}$$

(E.10)

$$\nabla\psi(\omega)^T d = \frac{\zeta}{\nu(\omega)} \cdot (2F(\omega)^T(-F(\omega)) + e^T(-G(\omega)) + \sigma\mu(\omega)n_2) + n_2 - \sigma\mu(\omega)e^T G(\omega)^{-1}$$

(E.11)

$$\nabla\psi(\omega)^T d = -\frac{\zeta}{\nu(\omega)} \cdot (2F(\omega)^T F(\omega) + (1-\sigma)e^T G(\omega)) + n_2 - \sigma n_2 \frac{e^T G(\omega)}{n_2} \cdot \frac{e^T G(\omega)^{-1}}{n_2}$$

$$\nabla\psi(\omega)^T d \le -\zeta(1-\sigma) + n_2 - \sigma n_2 \left[\prod_{i=1}^{n_2} G_i(\omega)\right]^{1/n_2} \left[\prod_{i=1}^{n_2} G_i(\omega)^{-1}\right]^{1/n_2} \qquad \textbf{(E.12)}$$

$$\nabla\psi(\omega)^T d \le -\zeta(1-\sigma) + n_2(1-\sigma) \qquad \textbf{(E.13)}$$

$$\nabla\psi(\omega)^T d \le -(1-\sigma) + (\zeta - n_2) \quad \textbf{(E.14)}$$

where the inequality in (E.12) follows from the arithmetic-geometric mean inequality applied to the terms $e^T G(\omega)/n_2$ and $e^T G(\omega)^{-1}/n_2$ and the fact that $2 > (1-\sigma)$.

Finally, Wang establishes the last assertion in *(iii)*. Using the second assumption and the fact that Ω_Σ is an open set, it is concluded that there exists a scalar $\bar{\lambda}$>0 such that, for all $\lambda \in (0,\bar{\lambda})$, $\omega + \lambda d \in \Omega_\Sigma$ and

$$\psi(\omega + \lambda d) - \psi(\omega) \le \lambda(\nabla\psi(\omega)^T d + (1-\alpha)(1-\rho)(\zeta - n_2)) \quad \textbf{(E.15)}$$

$$\psi(\omega + \lambda d) - \psi(\omega) \le -\alpha\lambda(1-\sigma)(\zeta - n_2) \quad \textbf{(E.16)}$$

Q.E.D.