

Chapter 2

Mathematical Modeling Using Algebraic Oriented Languages for Nonlinear Optimization

In the last two decades, significant research activities have taken place in the area of local and global optimization, including many theoretical, computational, and software contributions. The access to this advanced optimization software needs more and more sophisticated modeling tools. Algebraic oriented optimization modeling languages represent an important class of tools that facilitate the communication of optimization models to decision-making systems based on the optimization paradigm. In a wider context, the algebraic oriented modeling tools evolve toward fully integrated modeling and optimization management systems with access to databases, spreadsheets, and graphical user interfaces.

As already known, at the fundamentals of every mathematical model lie the conservation laws which are active in the domain to which the mathematical model belongs. Therefore, knowing these conservation laws is essential, and their usage leads to models satisfying the *adequacy to real principle*. However, besides the conservation laws, additional empirical knowledge, principles, different rules, or previous experience must be considered in order to develop a mathematical model of a given reality.

In this chapter, certain aspects concerning the mathematical modeling process in the context of mathematical modeling technologies based on the algebraic oriented languages are discussed.

2.1 Linguistic Models Versus Mathematical Models

In a common way, everyone is familiar with different representations of the surrounding world, the so-called mental models, used in each moment of their existence. The decisions made in different situations are not based on the real world but on our mental images of the real world, on our mental images of the relations among the components of the real world. The mental models represent our

understanding of the part of the creation that we want to know. Having in view that the support of our thinking is the word, it follows that mental models are actually linguistic models. Therefore, in one way or another, the human beings set forth their understanding of the real world as a linguistic description expressed as a corpus of assertions (theorems) of the general form: *if ... then ...*.

The mental models have some advantages which recommend them to be used in our efforts to understand the world. A mental model is *flexible* in the sense that it can take into consideration a domain of information sensibly larger than the numerical one. Besides, a linguistic model may be quickly *adapted* to some new situations and can be *modified* as soon as some new information is available. The mental models are *filters* through which we can explain our experiences, evaluate, and select different actions. In a way, the greatest philosophical systems, political and economical doctrines, theories of physics, and literature itself are linguistic models.

However, the mental models have some disadvantages as well. They are not so easily understood by the others. Their interpretation is *dependent* on their creator. Besides, the hypothesis used to generate them is difficult to examine and even to accept. The ambiguities and contradictions contained in these types of models can remain undetected, unexplained, and therefore unsolved. It is quite natural to have difficulties in understanding the linguistic models suggested by the others. Surprisingly, we are not so good at developing and understanding our own mental models or at using them during the process of decision-making. Psychologists have shown that we are able to consider only a very small number of factors in decision-making. In other words, the mental models we use in decision-making are *extremely simple*. These are often imperfect because we frequently persist in error when deducing the consequences from the suppositions on which they are based. These models often express what we would like to happen and not what actually happens in reality.

But the greatest *imperfection* of the mental models lies in the fact that these intellectual developments do not satisfy the criteria of *completeness*, *minimality*, and *non-contradiction*. It is very likely that some very important assertions which change their significance may be omitted in a mental (linguistic) model. Clearly, some new assertions may be introduced, which are in contradiction with those we have previously considered in our reasoning. At the same time, in using the linguistic models, we are often faced with the very difficult problem of *word rectification*. This problem dramatically limits the usage of linguistic models with different groups of people. Finally, we must emphasize that during the linguistic model analysis and solving process, we are very likely to be confronted with the *circularity danger*. The problem of the circularity of formal systems was solved by Gödel (1931), who showed that expressing knowledge into a formal system and into logical formal systems, like those of Russell or of Zermelo-Fraenkel-Newmann, is an illusion. There are relatively simple assertions (theorems) which are impossible to be solved (decidable) in this kind of formal systems.

A mathematical model is a representation in mathematical symbols of the relations between the variables and the parameters belonging to the part of the creation we are interested in. The relations describing the mathematical model

often include variables and their derivatives, thus expressing the *local character* of the model and that of *predictability* as well. The mathematical models have a number of advantages versus the linguistic models. The most important is that mathematical models do not have any imperfections which the linguistic models do. They are explicit in the sense that the hypothesis and the assumptions used in their development are public and subject to any criticism. Besides, the (logical) consequences after solving them are very well justified mathematically. Finally, the mathematical models are more comprehensive, being able to simultaneously consider an appreciable multitude of factors. But the most important characteristic of mathematical models is that they are written on the basis of the conservation laws. In this respect the Noether theorem shows that the conservation laws are direct consequences of different symmetries (Andrei, 2008b). For example, the conservation of energy is a consequence of temporal symmetry, while the conservation of momentum is a consequence of the symmetry of space. The entire physical world is depicted as being governed according to mathematical laws.

2.2 Mathematical Modeling and Computational Sciences

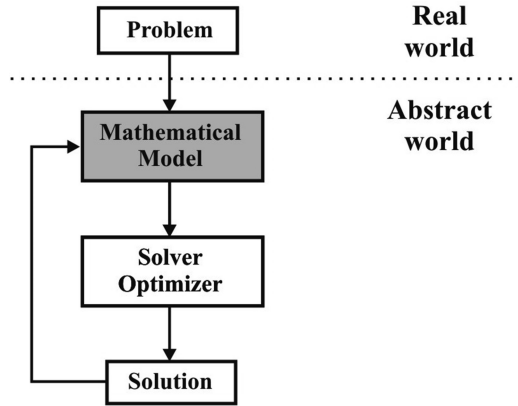
The theory and practice of mathematical modeling is based on computational sciences. Broadly, computational sciences consist in the usage of computer systems for analyzing and solving scientific problems. There is a sharp distinction between computer science and computational science. Computer science focuses on the building and the analysis of computers as well as on computability. On the other side, computational sciences emphasize the development of methods, techniques, and algorithms for solving mathematical models and centers on the convergence and complexity of algorithms.

The purpose of computational sciences is to understand the evolution of a real-world system by analyzing and solving the corresponding mathematical models associated to the considered system, using high-performance computers. By using mathematical symbols as well as mathematical accuracy and rigor, mathematical modeling has a central place in computational sciences. Mathematical modeling leads us to the maturation of the domain the mathematical model belongs to and also to the generation and the development of new mathematical methods and techniques able to solve new problems. The place of the mathematical model within the general scheme for modeling and solving a problem is shown in Figure 2.1.

Observe that a mathematical model is synthesized from the real world, thus arriving into an abstract world, where mathematical rigor is present through mathematical concepts. The model is solved using mathematical theories irrespective of the physical significance of the elements of the problem. The interpretation of the solution may lead us to some modifications of the model, which again determines our addressing to the real world.

Notice that in the abstract world of the modeler, we operate with concepts belonging to the domain the modeling process refers to. This is the question of

Figure 2.1 The process of modeling and solving a problem



the phenomenology and the conservation laws which characterize the domain of interest. At the same time, certain concepts from the theory of languages, of translators, and of compilers are used in order to develop informatics technologies able to elaborate, update, and maintain the mathematical models.

On the other hand, in the abstract world of algorithm developers, one operates with advanced mathematical concepts in order to generate optimization algorithms and studies of their convergence and complexity. Additionally, some advanced languages like Fortran or C++ are used here for implementing algorithms in computer programs.

It is quite clear that for the majority of problems, their mathematical models must be simplified. Therefore, we are faced with the following dilemma. *What is best: to consider an approximate (simplified) mathematical model of the problem and then try to get an exact solution as much as possible or to use a mathematical model as accurately as possible and then determine an approximate solution of it?* The practical experiences recommend that the second alternative gives better results.

2.3 Modeling Scheme

Formulating and developing a mathematical model of a given reality is one of the finest creative intellectual activities. A deep analysis of the modeling and solving process (see Figure 2.1) shows that mathematical modeling involves more effort and time for data analysis, verification, and documentation, as well as for updating and improving different variants of the model. The difficulties of the data management for mathematical modeling are coming from the fact that nowadays we do not have a clear methodology for mathematical modeling expressed as an algorithm or as a group of algorithms to be used in a general context. However, understanding

the linguistic representation of the process that we need to represent through a mathematical model is crucial.

The development of a mathematical model is a complex process consisting in a mapping between the objects (together with the relations among them) from the real-world and the symbolic and mathematical objects. This process involves both a very good knowledge of the reality we want to represent in mathematical terms and a multitude of methods and techniques for mathematical modeling. The correctness of a model is usually established after its solving. As a result of intensive computational experiments, it is often necessary to reformulate the model by introducing some new algebraic or differential equations (which were initially ignored) and also the corresponding data. The mathematical modeling process is closely linked to the solving process of the model, supporting each other. These two processes interact in order to build up a mathematical object placed in the *perspective of the infinite similarity with reality*.

Let us now try to detail the modeling process. At the very beginning, for the process we want to represent into mathematical terms we have a *linguistic description*, actually a text expressed in the natural language which describes the process. It is worth saying that this linguistic description is a *scientific text*, not a literary one. Obviously, this description includes references to the important conservation laws which are active in the process domain under consideration. As a result of an abstracting creation activity from this linguistic description and using the conservation laws, we get an *algebraic-differential representation* of the process, a *mathematical model*. This activity mainly consists in identifying the variables and parameters of the process, together with the algebraic and differential relations among these entities by using principles, empirical knowledge, different rules, and some other additional knowledge of the process. Once having this algebraic-differential representation of the process, the next step is to develop an *internal representation* of it, a representation directly admitted by any professional solver, in our case an optimizer. This is done through the *external representation* of the model using the so-called algebraic oriented mathematical modeling languages. Therefore, the mathematical model has three forms of representation: an algebraic-differential one using mathematical symbols, an external one using an algebraic oriented mathematical modeling language (like GAMS, AMPL, ALLO, etc.), and an internal representation which is mainly an informatics description including different files and databases directly admitted by any solver (optimizer). Figure 2.2 presents the modeling scheme for nonlinear optimization based on algebraic oriented languages.

Suppose we have written the optimization mathematical model, i.e., we have specified the objective function, the constraints, and the simple bounds on variables. The problem now is how to transmit this model to the solver (optimizer) in order to get a solution. Since we consider mathematical models with thousands of variables and constraints, this problem appears to be quite challenging. An elegant and very efficient solution is to use algebraic oriented languages for achieving an external description of the model which can automatically be translated into the internal description of it by means of a translator associated to the language used in this

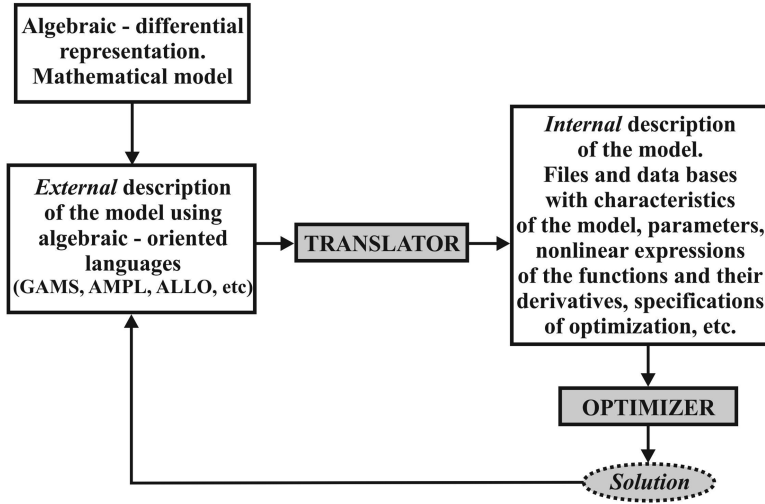


Figure 2.2 Modeling scheme of nonlinear optimization based on algebraic oriented languages

respect. For example, in case of linear programming problems, the internal representation of the model is given by the MPS form. Therefore, the translator associated to an algebraic oriented language (like GAMS, AMPL, ALLO, etc.) generates the MPS form of the model from its external description. In case of nonlinear optimization problems, the mechanism is more complex but mainly the same.

2.4 Algebraic Oriented Modeling Languages

Algebraic modeling languages represent a problem in a purely declarative way. Most of them include some computational facilities to manipulate the data as well as certain control structures. Many optimization models can be well represented declaratively in a very concise way, which also favors the insight of the model. Besides, a mathematical model is stated in a declarative way using mathematical equalities and inequalities. This gives a clear documentation of the model which can be parsed and interpreted by a compiler.

The algebraic languages are more declarative rather than imperative. A declarative language emphasizes *what is being computed* more than *how it is being computed*. Furthermore, the reasons for which we use a declarative presentation of models are *conciseness*, *insight*, and *documentation*.

It is important to emphasize that one of their strengths is the complete separation of the problem formulation from finding a solution which is supposed to be determined by an external program called *solver* or *optimizer*. This characteristic allows the modeler not only to separate the two main tasks of model formulation

and model solving but also to easily switch among several solvers. This is very important indeed because it is known that a model instance can be solved using one method, while another instance is efficiently solvable by using another method. Another important advantage of algebraic oriented modeling languages is that they clearly separate between the model structure which only contains parameters, but not data, and the model instance in which the parameters are replaced by a specific set of data.

Therefore, the main features of the algebraic oriented modeling languages are:

- Purely declarative representation of the model
- Clear separation between formulation of the model and its solving
- Clear separation between the model structure and the model data

An algebraic oriented modeling language has a declarative and an algorithmic knowledge which are clearly separated. Either of them may be empty, meaning that the problem is represented as a purely declarative or as a purely algorithmic form. The declarative part consists of the basic building blocks of declarative knowledge including variables, parameters, constraints, sets, etc. On the other hand, the algorithmic part consists of the control structures describing (explicitly or implicitly) the computation of solving a problem.

The most important algebraic oriented languages used for solving nonlinear optimization problems are presented in Table 2.1.

What is characteristic of algebraic oriented languages is that they permit to the modeler to express the model in a way based on indices that belong to certain abstract entities, which are sets of evolution of indices, parameters, variables, and constraints. Furthermore, there is the possibility of grouping the entities in sets with similar characteristics to which we can make reference through indices as elements of certain sets. The grouping of entities (such as variables or constraints) allows their representation in a compact manner, very similar to the algebraic one. For instance, the mathematical formulation $\sum_{i \in I} x_i$ can be translated into the expression **SUM(I, x(I))** in the GAMS modeling language. Therefore, as can be seen, this leads to a problem formulation that is very close to the formulation using algebraic notations. The task of the translator associated to the modeling language is to expand this compact representation into one accessible to a solver, the so-called the *problem instantiation*, the one ready to be considered by an appropriate optimization solver. This expanding operation is realized by replicating every entity over the different elements of the set. This is referred to as a *set-indexing* ability of the algebraic oriented modeling languages. The modeler can define generic expressions that are indexed over several sets. Set indexing in such cases involves compound sets.

As already mentioned, the algebraic oriented modeling languages use essentially declarative statements as opposed to programming languages (Fortran, C++, etc.) in which the use of procedural statements dominates over the use of declarative ones. For instance, the few procedural statements used in algebraic oriented languages are **read/write data** and **solve** commands. In order to deal with more complicated nonlinear optimization models, these languages include **if-then-else**,

Table 2.1 Algebraic oriented modeling languages (partial list)

GAMS	<p><i>General Algebraic Modeling System</i> (www.gams.com) Development Research Center. The World Bank, 1818 H. Street, Washington D.C., USA. Brooke, A., Kendrick, D., Meeraus, A., Raman, R., Rosenthal, R.E., <i>GAMS A user's guide</i>. GAMS Development Corporation, December 1998</p>
AMPL	<p><i>A Mathematical Programming Language</i> (www.ampl.com) Department of Industrial Engineering and Management Sciences. Northwestern University, Evanstone, Illinois 60,201, USA. Fourer, R., Gay, M., Kernighan, B.W., <i>AMPL: A modeling language for mathematical programming</i>. Second edition. Duxbury Press/ Brooks/Cole Publishing Company, 2002</p>
ALLO	<p><i>A Language for Linear Optimization</i> (http://camo.ici.ro/projects/allo/allo.htm) Research Institute for Informatics – Bucharest. 8–10 Bdl. Măreșal Alexandru Averescu, sector 1, 011455 Bucharest – Romania. Andrei, N., <i>The ALLO language for linear programming</i>. ICI Technical Report No.1/2004, Bucharest. Andrei, N., <i>Criticism of the linear programming algorithms reasoning</i>. Academy Publishing House, Bucharest 2011. (see Chapter 17, Annex A6)</p>
LPL	<p><i>A Structured Language for Modeling Linear Programs</i> (http://diuflx71.unifr.ch/lpl/mainmodel.html) Institut of Informatics, University of Fribourg, Regina Mundi, rue de Faucigny 2, CH-1700 Fribourg, Switzerland. Hürlimann, T., <i>The LPL modeling language: Highlights</i>. University of Fribourg, Switzerland, December 8, 2011</p>
AIMMS	<p><i>The Modeling System</i> (www.aimms.com) Paragon Decision Technology B.V. P.O. Boxes 3277, 2001 DG Haarlem, The Netherlands Bisschop, J., Roelofs, M., <i>AIMMS – The user's guide</i>. Paragon Decision Technology, 1999</p>
MPL	<p><i>Modeling System</i> (www.maximalsoftware.com) Maximal Software, Inc., 2111 Wilson Blvd. Suite 700, Arlington, VA, USA. Kristjansson, B., <i>MPL User manual</i>. Maximal Software Inc., Iceland, 1993</p>
LINDO	<p><i>Powerful Library of Optimization Solvers and Mathematical Programming Tools</i> (http://www.lindo.com) Lindo Systems Inc., 1415 North Dayton Street, Chicago, IL, 60,642, USA. Schrage, L., <i>Optimization Modeling with LINDO</i>. 5th edition. Duxbury Press, 1997</p>
MOSEL	<p><i>An Extensible Environment for Modeling and Programming Solutions</i> (www.fico.com/en/Company) FICO® Xpress Optimization Suite. 901 Marquette Avenue, Suite 3200, Minneapolis, MN 55402, USA. (formerly DASH) Colombani, Y., Heipcke, S., <i>Mosel: An Overview</i>, Xpress Team, FICO, Leamington Spa CV32 5YN, UK. (http://www.fico.com/xpress)</p>
TOMLAB	<p><i>TOMLAB Optimization Environment</i> (tomopt.com/tomlab) 113 Cherry St Ste. 95,594. Seattle, WA 98104–2205, USA</p>

for, while commands. Such commands enable the modeler to write certain solution algorithms directly in the modeling language (Bisschop & Meeraus, 1982; Fourer, 1983; Hürlimann, 1999; Kallrath & Wilson, 1997; Conejo, Castillo, Minguez, & Garcia-Bertrand, 2006; Castillo, Conejo, Pedregal, García, & Alguacil, 2001). The next chapter is dedicated to illustrate the main aspects and facilities of the GAMS algebraic oriented language.

Notes and References

This chapter on modeling for nonlinear optimization using algebraic oriented languages for continuous nonlinear optimization is based on developments presented in Andrei (2012, 2013b).

Continuous Nonlinear Optimization for Engineering
Applications in GAMS Technology

Andrei, N.

2017, XXIV, 506 p. 68 illus., 66 illus. in color., Hardcover

ISBN: 978-3-319-58355-6