DISCUSSION PAPER

DRD118

# MODELING SYSTEMS AND NONLINEAR PROGRAMMING IN A RESEARCH ENVIRONMENT

Anthony Brooke
Arne Drud
Alexander Meeraus

April 1985

Development Research Department
Economics and Research Staff
World Bank

# MODELING SYSTEMS AND NONLINEAR PROGRAMMING IN A RESEARCH ENVIRONMENT

by

Anthony Brooke
Arne Drud
Alexander Meeraus

April 1985

# MODELING SYSTEMS AND NONLINEAR PROGRAMMING IN A RESEARCH ENVIRONMENT

by

Anthony Brooke*
Arne Drud**
Alexander Meeraus**

Development Research ~~Department~~
World Bank

April 1985

* Consultant, The World Bank, supported by external research budget RPO 673-06

** Staff Members. The World Bank

Paper prepared for the American Society of Mechanical Engineers Conference on Computers in Engineering, Boston, Massachusetts, August 1985.

## ABSTRACT

Nonlinear programming (NLP) algorithms are often designed for expert users and users from other fields can find them hard to work with. The paper argues that High Level Modeling Systems can help researchers from other fields use NLP productively. It describes the traditional approach to NLP modeling, outlines its limitations, and shows with examples from a particular modeling system, the General Algebraic Modeling System (GAMS), how the limitations can be removed. Finally, the paper reports on the use of GAMS in a research environment.

Key words: Nonlinear Programming, Modeling Systems

# MODELING SYSTEMS AND NONLINEAR PROGRAMMING IN A RESEARCH ENVIRONMENT

## TABLE OF CONTENTS

## I. INTRODUCTION

Nonlinear Programming (NLP) is often thought of as consisting only of nonlinear programming algorithms, and much research has been devoted to NLP algorithms in recent years. If we are interested in the usefulness of NLP as a tool in other disciplines, however, it is not enough to look at the algorithms. We must also consider the many other constraints in the overall modeling process, i.e, the process of model building, model implementation, model solution, report writing, and model documentation. And we must realize that the constraints imposed by the algorithms, i.e, by the model solution step, are often less significant in practice than the constraints imposed by the other steps in the modeling process, in particular by the model implementation step.

These observations suggest that the usefulness of NLP can be greatly improved by improving the tools for implementing the other steps in the model building process, without changing the optimization algorithms themselves. The most appropriate tools for a given task will of course depend on the nature of the work environment; in this paper, we have distinguished between two environments with very different requirements. For the sake of simplicity, we call these environments the operational environment and the research environments.

In a typical operational environment, the same type of model is solved repeatedly. In this case, we can think of special purpose model implementation tools that take advantage of the special structure of the particular problem. A hypothetical example might be a system used to help design an oil refinery. Such a system could have 'knowledge' of how an oil refinery works; the user could define an overall model by defining a set of units from a prespecified set of well defined units and a network of pipes and valves connecting the units. The system would be able to construct a model, with process equations for the selected units and material balance equations for the network, and submit the model to an optimization algorithm for solution. The system would even be able to use established rules of thumb to select initial values for the optimization. The key to the operational environment is the availability of 'knowledge', that can be used to make the modeling tools user friendly and efficient.

The research environment, on the other hand, is characterized by constantly changing models and by the lack of a pool of standardized knowledge that can be packaged into the model building tools. In this instance, therefore, modeling tools cannot be designed to take advantage of any specific problem structure or body of knowledge. The tools for this environment must be general purpose; they can therefore only rely on a few things that are common to most modeling projects, such as the use of mathematics. Until recently, it was common practice to use FORTRAN as the general tool in the research environment; model were implemented using specially written FORTRAN programs. Lately, however, alternative 'High Level Modeling Systems' have appeared and this paper examines the potential advantages of such systems in the research environment, using examples from a particular system, the General Algebraic Modeling System (GAMS).

The discussion that follows is confined to the research environment. Section II describes the typical modeling process in this environment and explains why current techniques are inadequate. In addition to a general shortage of tools, this inadequacy stems from the fact that the input to NLP codes has usually been designed to be algorithm or machine friendly and little has been done to make it user friendly; moreover, different codes use very different types of input. The position is further complicated by the possibility that one code may prefer one type of mathematical model formulation, while another may prefer a different but mathematically equivalent formulation.

Sections III and IV show how GAMS, a high level modeling system developed at the World Bank, tries to improve on this situation. GAMS breaks the input phase into two parts. The first part, described in Section III, comprises the model formulation process itself and the definition of the associated data. GAMS uses a powerful and algorithm independent representation of model and data that allows the user to define large and complex models and to test them for logical consistency in a very short time. The second part of the input phase, discussed in Section IV, is the translation of the user's model representation into the input format of the NLP code. This is done automatically by GAMS once the user has chosen a NLP code. GAMS knows the input (and output) format of several NLP codes, and the user can switch from one to another by simply changing a code name. Section V ends the paper with a short description of experience to date with the GAMS modeling system.

## II. THE MODELING PROCESS

The process of building and using models, linear or nonlinear, is composed of five interrelated steps: model formulation, model implementation, model solution, report writing, and model documentation. The efficiency of the overall modeling process depends on the techniques used in each of the five steps, and on how well these techniques are integrated. We will try in the following paragraphs to cover each step and emphasize the links between steps (see also [1]).

A mathematician may think of a model simply as a set of variables tied together by constraints or equations. In practice, however, a set of data and the transformations that convert the data into model coefficients are equally important. We define the model formulation process as that of defining the basic data set, defining the transformations of the data that create the coefficients of the model, and defining the variables and the constraints. Note that our definition contains both a procedural or recursive part, the data transformations, and a non-procedural or simultaneous part, the variables and the constraints. It is important to realize that we are probably dealing with multidimensional quantities such as demand in different regions for different products, output of different products in different plants in different periods, stress in different points of a construction under different external load scenarios, etc. Basic data, derived coefficients, variables, and equations can all be multidimensional.

It is also important to realize that exceptions exist in all but the most trivial models: initial and terminal periods differ, some plants only produce some products, not all pairs of nodes in a network are connected, and so forth. This is equivalent to saying that the multidimensional quantities can be sparse. The model formulation process must therefore rely on techniques for defining data, data transformations, and variables and equations over multidimensional domains, subject to exceptions or sparsity. Conventional algebraic notation with subscripts and summations is well suited for this purpose, and most models are initially defined using this notation.

The subsequent process of implementing the model on a computer can be broken down into several linked subprocesses: structuring and entering the basic data, programming the data transformations, selecting the solution code, and translating the model into the format required by the solution code. The translation step will usually involve writing a FORTRAN subroutine that can compute the constraint values for given values of the variables. It sometimes involves writing subroutines for the derivatives of the constraints; for large models it may also involve specifying a sparsity pattern.

Using conventional techniques, the FORTRAN code will be written using a mixture of two approaches: the block approach and the single equation approach. The block approach exploits the multidimensional structure of the problem. The code will use multidimensional arrays and will be full of DO-loops, but it will also contain complicated constructs to handle the exceptions; these make the code difficult to read, verify, and modify. This approach was used with the first GAMS-NLP interfaces; further details can be found in [2]. The single equation approach eliminates all the complications of exceptions, by treating each equation and variable within a block as an individual equation or variable. The code becomes very simple, but also very long, and the structure of the underlying model vanishes. It is therefore just as difficult to read and verify as the block structured code, and just as difficult to modify. It is also much more expensive to compile.

Some NLP codes require more than a FORTRAN subroutine that evaluates constraints. They may also require a sparsity pattern and/or a subroutine that computes derivatives. This additional input is essentially a transformation of the information already built into the constraint subroutine. The user must thus provide the same information in two formats, and in a consistent manner.

Once the groundwork has been covered, the solution of the model itself is usually straightforward. It involves selecting some initial values and calling the NLP code. Difficulties arise only if the NLP code cannot solve the model, something that happens more frequently than modelers like to admit. There are only three remedies. The simplest one is to try with another set of initial values--for example, ones computed from the basic data. The two other approaches are to try a different NLP code, or to change the model formulation into an equivalent one--by scaling rows or columns, for example--and to try to solve the new model with the old NLP code. Both these alternatives can be very expensive since they involve redoing a great deal of labor intensive programming in the model implementation phase. We have mentioned that failures are common, but we should also note that most problems can be solved by an experienced modeler with one of the existing codes after a few initial points and a few formulations have been tried; thus, solutions can be obtained reasonably reliably if we are prepared to spend time and money.

After completion of the solution phase, we are now ready to produce a set of reports. Report writing involves accessing solution values, often both primal and dual, performing transformations of them, and writing tables with the transformed values. The transformations will often involve the original data set and/or some of the coefficients, so the report software must have access to all this data. The conventional technique is to write special purpose software for report writing, using either a block approach or a single equation approach as discussed above.

The last phase, documentation, should ideally be carried out in parallel with the other four. We usually think of documentation in terms of two items, the documentation of the mathematics of the model, and the documentation of how the model has been implemented on a computer. Recent papers, e.g., Gass, [7], discuss the documentation aspects of modeling, using some very expensive modeling exercises as examples. These papers implicitly assume that conventional techniques were used for implementing the model; under these circumstances, the issue of documentation becomes largely a matter of documenting the FORTRAN code, and of setting up mechanisms to ensure that model and program documentation are updated in parallel with the FORTRAN program to ensure that they represent the same model.

Thus all the work involved in conventional NLP modeling, except the solution process itself, relies on software specially developed for the model at hand, making the modeling process very labor intensive. It also becomes very inflexible because there is no good way of rewriting code. In addition, there is a significant probability that the model defined through our FORTRAN code differs from the model we think we have defined because of undetected errors.

We believe that NLP modeling will only become widely used if current technology can be made cheaper and more reliable. This requires that the modeler should only need to enter a certain piece of information into the computer once, and that he/she should be able to enter it in a format that is natural to him/her. The computer should then transform information as needed. Such a procedure has several advantages: it improves the productivity of the modeler by avoiding repetitions, it eliminates many transformation errors caused by manual operation, and it makes documentation much easier. It also eliminates the documentation of FORTRAN code, and can make the computer input self documenting.

We call a computer program that accepts user friendly model definitions and performs all the transformations needed to interface with a solution code a modeling system. The next section decribes one such modeling system, GAMS, and shows how it can solve a number of the problems outlined above.

## III. GAMS AND THE MODELING PROCESS

This section briefly describes some of the key features of the GAMS language. The description is intended to illustrate with examples some of the most important features of GAMS and how they can increase the productivity of modeling; the reader who is interested in further details is referred to Kendrick and Meeraus [10] and Meeraus [12]. The examples are drawn from a small water distribution model but the techniques are generally applicable. The overall GAMS representation of the model is shown at the end of the paper.

Models become large not because they contain many different concepts, but because the concepts they describe are repetitive and defined over large sets, e.g., demand at many locations or production in many plants. It is therefore important that a modeling system has facilities for defining sets, and for defining data, data transformations, variables and equations over sets or cartesian products of sets. Below are two examples of GAMS set definitions.

```
SET N     NODES          /NW, E, CC, W, SW, S, SE, N/
    RN(N) RESERVOIRS     /NW, E/;
```

N and RN are the names of the sets NW, E, CC, W, SW, S, SE, and N are the elements of the set N and NW and E are the elements of RN. The strings NODES and RESERVOIRS are (optional) documentation text. The (N) following RN indicates that RN belongs to the domain of N, i.e., is a subset of N. GAMS will check that the elements in the definition as well as in later uses of RN do in fact belong to N, thereby preventing both spelling errors and logical model building errors.

Data can be entered in simple tabular form:

```
TABLE NODE(N,*) NODE DATA

          DEMAND       HEIGHT        X        Y

NW                      6.50       1700     3600
W          0.452        5.16        750     2400
CC         1.212        3.02       2000     2300
E                       3.25       4000     2200
```

The (N, *) following NODE indicates that NODE is two dimensional, that the first dimension must be a check against the domain N, and that the second dimension should not be checked. Note that sets in GAMS are deliberately unordered; the elements of set N are ordered differently in the SET statement and the TABLE statement, but GAMS interprets the data by matching the labels and not the positions.

Once the basic data have been defined with SET and TABLE statements, it is easy to transform them into the coefficients needed in the model. A PARAMETER statements identifies a collection of exogenous data, and assignment statements can be used to define their values.

```
PARAMETER DIST(N,N) DISTANCE BETWEEN NODES;
ALIAS (N,NP);
DIST(N,NP) = SQRT(SQR(NODE(N,"X")-NODE(NP,"X"))
           +  (SQR(NODE(N,"Y")-NODE(NP,"Y")));
```

The assignment statement defines DIST for all pairs of nodes. The ALIAS is used to define a set NP that has the same elements as N but otherwise is independent. If we had written DIST(N,N)=.. we would only have generated the diagonal of the square matrix DIST.

As well as the conventional algebraic operations, GAMS defines set operations such as unions and intersections, and operators over cartesian products like SUM, PROD, MIN, and MAX, making it easy to define aggregations and matrix products, and to perform other frequently used transformations.

We can now use the basic data and the derived coefficients to define the variables and equations that make up the model. In the example below we will show a set of flow conservation equations.

```
VARIABLES  Q(N,N)        FLOW BETWEEN NODES
           S(N)          SUPPLY AT RESERVOIRS

EQUATIONS  CONT(N)       FLOW CONSERVATION;

CONT(N)..  SUM(NP,Q(NP,N) - Q(N,NP)) + S(N) =E= NODE(N,"DEMAND");
```

The careful reader will have noticed that the above equation assumes that all pairs of nodes are connected and that there is supply in all nodes. In practice, we must remove Q(N,NP) for all pairs of nodes that are not connected, and S(N) for all non-reservoir nodes. This and other types of exceptions frequently occur in large models, and a consistent and convenient way of representing them is crucial in any modeling system.

The first step in our application is to define a correspondence (a sparse two-dimensional set), that defines the arcs of the network.

```
SET A(N,N) NETWORK ARCS ARBITRARILY DIRECTED
    /NW.(W,CC,N), E.(N,CC,S,SE), CC.(W,SW,S,N), S.SE, S.SW, SW.W/
```

An arc is represented by an ordered pair of nodes, e.g., S.SE, and NW.(W,CC,N) is a shorthand for NW.W, NW.CC, NW.N . Exceptions or sparsity are now defined with a $ or "such that" operator. Applied to the CONT equation above it yields

```
CONT(N)..   SUM(NP, Q(NP,N)$A(NP,N)-Q(N,NP)$A(N,NP))
            + S(N)$RN(N) =E= NODE(N,"DEMAND");
```

The \$A(NP,N) means that the term Q(NP,N) is only included in the summation if A(NP,N) is defined which means an arc existed. Similarly, S(N) is included only if the current element of N belongs to RN, the set of reservoir nodes, meaning there is supply only from reservoirs. The \$-operator can also be applied to the domain over which an equation or an assignment statement is defined, as in the following pressure loss equation.

```
VARIABLE   H(N)            PRESSURE IN EACH NODE

EQUATION   LOSS(N,N)       PRESSURE LOSS EQUATION;

LOSS(N,NP)$A(N,NP)..       H(N) - H(NP) =E=expression;
```

Exceptions can also be handled through the use of subsets, as in the following pumping cost equation.

```
          PARAMETER  PC(RN)         PUMPING COST PARAMETER
          VARIABLE   PCOST          TOTAL PUMPING COSTS
          EQUATION   PEQ            PUMPING COST EQUATION

  PEQ..   PCOST =E= SUM(RN,S(RN) + PC(RN)*(H(RN)-NODE(RN,"HEIGHT")));
```

Notice that H is defined over the set of all nodes, but only referenced over the set of reservoir nodes. This does not create any conflict; GAMS matches set elements properly.

A model in GAMS is defined as a collection of equations and the variables they contain. To start optimizing, we simply state the objective, its direction, and the generic problem class. Particular codes can be selected using OPTION statements.

```
MODEL NETWORK / CONT, LOSS, PEQ, other equation names /;
OPTION NLP = nlp-code;
SOLVE NETWORK USING NLP MINIMIZING objective;
```

Notice that the definition of the model is completely independent of the solution code, The solution code is only mentioned in the OPTION statement above. This is very important; if the solution attempt fails with one algorithm, it is easy to switch to another by simply changing this one statement.

The execution of the SOLVE statement entails several verification and testing steps. GAMS checks that all sets, parameters, and equations used in the model have been defined and it checks that the model belongs to the class of models that can be handled by the particular NLP code. The restrictions are that equations must be differentiable functions of the variables and that variables cannot be binary. The user can disable the differentiability test by defining the model as a discontinous NLP (DNLP), but this must be done explicitly and should only be undertaken by experienced users. Then, if all is well, the model implementation and solution steps outlined in section II are executed. GAMS will create the problem representation in the correct format, possibly create and compile a FORTRAN subroutine, design a solution strategy, and transfer control to the NLP code. After the solution code has terminated, GAMS will read the output and translate it back into GAMS notation for storage in the GAMS data base. Normally the user will not see any output from the NLP code. Only if it fails in an unexpected way will code specific output be returned to the user.

The report writing step is straightforward. The solution values were saved in the GAMS database by the SOLVE statement, and they can now be used in further data manipulations, often using the DISPLAY statement to design and print tables. Four quantities are saved in the database for each variable: the level value, the lower bound, the upper bound, and the marginal value; four similar quantities are saved for each equation. The four values of variable Q can be referred to as Q.L, Q.LO, Q.UP, and Q.M, respectively. The level values of Q are displayed with a "DISPLAY Q.L;" statement yielding the printout in Fig. 1.

References to the four quantities associated with a variable or an equation can be on either the right or left hand side of an assignment, i.e., H.LO can be used to compute values for a report after a model has been solved, but it can also be assigned a value, e.g.,

**H.LO(N) = NODE(N,"HEIGHT") + 7.5 + 5.0*NODE(N,"DEMAND");**

Assignments of level values are used to define initial values for an optimization, and all the data manipulation machinery of GAMS is available. It is therefore easy to compute good initial values and to try alternative sets of these values. The use of the current level values as initial values for an optimization has an added advantage: if we want to solve a second, related model, GAMS will automatically use the optimal values from the first model as initial values for the second. It is sometimes useful to solve one

```
DESIGN OF A WATER DISTRIBUTION NETWORK   (WATER,SEQ=68)
E X E C U T I N G

----     102 VAR.L      Q          FLOW ON EACH ARC - SIGNED         (M**3 PER SEC)

             CC          W          SW          S          SE            N

NW        1.238      0.685                                           0.006
E         0.005                              0.644      0.244        0.447
CC                   0.007      0.010        0.011                   0.003
SW                  -0.240
S                              -0.005                   0.008
```

Figure 1: <u>Printout Produced by the "DISPLAY Q.L;" Statement</u>

or more relatively simple problems, perhaps linear ones, to find a good starting point from which the NLP can take over automatically. The use of current level values as inital values is also extremely useful if an NLP code cannot solve a model. It is easy to try to solve the model with another code, starting from the intermediate solution found by the first code. All the user has to do is to specify the name of the new code, and GAMS can take care of the rest. Automatic use of level values can also be useful after reformulations of certain parts of a model. The values of the variables will automatically be used by GAMS as long as their names are the same.

Documentation is the final step in the modelling process. The only thing that has to be documented is the GAMS source input; all other parts of the model are temporary data structures and can be ignored. GAMS has been designed to encourage the modeler to document the model as it is developed. All sets, parameters, variables, and equations can have documentation texts associated with them, and the documentation is kept in the GAMS database and displayed each time they are printed. The GAMS source input can be used as the complete documentation of the model, especially if the modeler makes a conscientious effort to make source text clear and easy to understand. The development of a model in GAMS is similar to that of writing a scientific paper: several drafts are usually needed before the message has been presented in a clear and concise way to make it as easy as possible for the reader. Several models developed in the World Bank now use the GAMS input as the final model documentation and include it in the final reports, as in for example Brown et al, [3], Choksi et al, [4], and Kendrick et al [9].

## IV THE GAMS INTERFACE TO NLP CODES.

One of the design philosophies of GAMS is that models should be represented in an algorithmically independent way. This is, of course, useful only if links exist to many different optimization codes, and preferably to codes that are built around different principles. At present, GAMS can communicate with four LP systems and four NLP codes, and more links, both generic and specific, will be added.

Table 1 gives an overview of several characteristics of the four NLP codes that can be used with GAMS. The codes are GRG 2 (Lasdon et.al. [11]), NPSOL (Gill et.al., [8]), MINOS 5.0 (Murtagh and Saunders, [14]), and CONOPT (Drud, [6]). The table entries describe different dimensions of an interface; the table has been included to show the kind of effort that must be put into a set of interfaces between a modeling system and several algorithms. A more technical description of how the interfaces have actually been built can be found in the appendix of Brooke et.al. [2].

## V. EXPERIENCE WITH GAMS.

GAMS has been available for LP models since 1980, and for NLPs since 1982. It has been used in the World Bank and at collaborating research and teaching institutions by people with widely differing experience in mathematical programming, mostly for solving problems in policy analysis, economics, and engineering. The benefits to these users have been substantial. For example, the modeler is more productive; in many cases, a modeler has been able to avoid employing a programmer to translate algebra into the machine representation, thus maintaining direct control of the project while still producing timely results. Perhaps more importantly, the details of a modeling

Table 1: SOME CHARACTERISTICS OF THE FOUR NLP CODES CURRENTLY ACCESSIBLE THROUGH GAMS

| Code | GRG2 | CONOPT | NPSOL | MINOS |
|---|---|---|---|---|
| Characteristics: | | | | |
| Algorithm Type: 1/ | GRG | GRG | SQP | PAL |
| Jacobian representation: | | | | |
| Sparse | No | Yes | No | Yes |
| Equations ordered | No | No | Yes | Yes |
| Variables ordered | No | No | No | Yes |
| Non linear elements distinguished | No | Yes | No 2/ | No 2/ |
| Subroutines: | | | | |
| Constraints and objective separated | No | No | Yes | Yes |
| Linear terms in functions | Yes | 3/ | Yes | No |
| Numerical derivatives | Yes | Yes | No | Yes |
| Numerical derivatives computed sparse 4/ | -- | Yes | -- | No |
| Name to index mappings 5/ | -- | Yes | -- | No |
| Miscellaneous | | | | |
| Stand-alone system | Yes | Yes | No | Yes |
| System can be modified | Yes | No | Yes | Yes |
| System controls memory, error recovery and file buffers | No | Yes | No | No |

Notes:

1. The algorithm types are: GRG - Generalized Reduced Gradient, SQP - Sequential Quadratic Programming, and PAL - Projected Augmented Lagrangian.
2. NPSOL considers all Jacobian elements in the nonlinear equations to be nonlinear, and MINOS considers all Jacobian elements occuring in both nonlinear variables and nonlinear equations to be nonlinear.
3. Linear terms can either be included in the FORTRAN subroutine or the coefficient can be defined through the MPS file.
4. Numerical derivatives are computed 'sparse' if several columns of the Jacobian can be computed in one call of the constraint subroutine (see Coleman and More, [4]).
5. Both MINOS and CONOPT define a sparse Jacobian through an MPS-type file, i.e., through names of variables. The subroutines use variable indices. CONOPT allows explicit mappings from names to indices, MINOS does not.

effort are now more readily accessible to others, and the traditional question about 'what really happens in this model' can be answered.

GAMS has also been found very useful in teaching. Models set up and solved by students have typically been small, because understanding and implementing the data structure for complex models is a tedious and specialized task. GAMS and its associated library of models permit students to concentrate on the model and not on the details of its implementation, and to experiment with formulation or data changes on models large enough to be interesting.

As mentioned in the introduction, GAMS has been designed to be a research tool. Once a problem is well understood through model experiments and the results are ready to be used operationally, it may be advantageous to switch to a modeling system that is more directly oriented towards the needs of the operational environment. A major oil company has used GAMS in its research department for some years without changing its operational modeling system. The company saw a need for experimentation with alternative model formulations before a final model was selected for operational use. The operational modeling system was not considered suitable for experimentation, and GAMS was selected for this task. After a robust formulation was found with GAMS, however, the model was transferred into the existing operational modeling system. The transfer was done manually in this case, but models can often be moved to other systems automatically by the same mechanisms as those used in the interfaces to model solution algorithms.

Finally, it is worth noting some usage statistics. In the World Bank alone, 15 projects currently use GAMS. Non-linear problems, ranging up to 700 equations, now outnumber linear problems. The GAMS system is accessed by World Bank users about 50 times on a normal weekday.

## REFERENCES

[1] J. Bisschop and A. Meeraus, "On the development of a General Algebraic Modeling System in a Strategic Planning Environment", Mathematical Programming Study, Vol. 20 (1982) pp. 1-29.

[2] A. Brooke, A. Drud, and A. Meeraus, "High Level Modeling Systems and Nonlinear Programming", Development Research Department, Discussion paper No.113, World Bank (1984) 30p.

[3] M. Brown, A. Dammert, A. Meeraus, and A. Stoutjesdijk, "Worldwide Investment Analysis, The Case of Aluminum", World Bank Staff Working Paper, 603 (1983) 168p.

[4] A. M. Choksi, A. Meeraus, and A. Stoutjesdijk "The Planning of Investment Programs in the Fertilizer Industry", John Hopkins University Press, (1980) 333p.

[5] C. F. Coleman and J. J. More, "Estimation of Sparse Jacobian Matrices and Graph Coloring Problems", SIAM Journal of Numerical Analysis, vol. 20 (1983) pp.187-209.

[6] A. Drud, "CONOPT - A GRG Code for Large Sparse Dynamic Nonlinear Optimization Problems", Mathematical Programming, vol.31, 1985, pp.153-191.

[7] S. I. Gass, K. L. Hoffman, R. H. F. Jackson, L. S. Joel, and P. B. Saunders "Documentation for a Model: A Hierarchical Approach" Communications of the ACM, vol.24 (1982), pp. 728-733.

[8] E. P. Gill, W. Murray, M. A. Saunders, and M. H. Wright, "User's Guide for SOL/NPSOL: A Fortran Package for Nonlinear Programming", Department of Operations Research, Stanford University, (1984) 34p.

[9] D. Kendrick, A. Meeraus, and J. Alatorre, "The Planning of Investment Programs in the Steel Industry", John Hopkins University Press, (1984) 310p.

[10] D. Kendrick and A. Meeraus, "GAMS, An Introduction", mimeo, Development Research Department, World Bank, (1984).

[11] L. S. Lasdon, A. D. Waren, A. Jain, and M. Ratner, "Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Programming", ACM Transactions on Mathematical Software, vol. 4 (1978), pp. 34-50.

[12] A. Meeraus, "General Algebraic Modeling System (GAMS), User's Guide version 1.0", Development Research Center, World Bank, (1982).

[13] A. Meeraus, "An Algebraic Approach to Modeling", Journal of Economic Dynamics and Control, vol. 5 (1983) pp. 81-108.

[14] B. A. Murtagh and M. A. Saunders, "A Projected Lagrangian Algorithm and its Implementation for Sparse Nonlinear Constraints", Mathematical Programming Study vol. 16 (1982), pp. 84-117.

# APPENDIX A

## GAMS LISTING OF EXAMPLE MODEL

```
DESIGN OF A WATER DISTRIBUTION NETWORK  (WATER,SEQ=68)                                        04/10/85 10:52:43  PAGE       1
                                                                                              GAMS 2.00 IBM CMS

  *  THIS EXAMPLE ILLUSTRATES THE USE OF NONLINEAR PROGRAMMING IN THE DESIGN OF
  *  WATER DISTRIBUTION SYSTEMS, THE MODEL CAPTURES THE MAIN FEATURES OF AN
  *  ACTUAL APPLICATION FOR A CITY IN INDONESIA.
  *
  *  REFERENCES:  A BROOKE, A DRUD, AND A MEERAUS, MODELING SYSTEMS AND
  *                    NONLINEAR PROGRAMMING IN A RESEARCH ENVIRONMENT,
  *                    DRD DISCUSSION PAPER 118, DEVELOPMENT RESEARCH DEPARTMENT,
  *                    WORLD BANK, 1985.
  *               A DRUD AND A AF ROSENBORG, DIMENSIONING WATER DISTRIBUTION
  *                    NETWORKS, M. SC. DISSERTATION, (IN DANISH), INSTITUTE OF
  *                    MATHEMATICAL STATISTICS AND OPERATIONS RESEARCH, TECHNICAL
  *                    UNIVERSITY OF DENMARK, 1973.
  *
 SET   N        NODES /  NW    NORTH WEST RESERVOIR
                         E     EAST RESERVOIR
                         CC    CENTRAL CITY
                         W     WEST
                         SW    SOUTH WEST
                         S     SOUTH
                         SE    SOUTH EAST
                         N     NORTH         /
     A(N,N)  ARCS (ARBITRARILY DIRECTED)  / NW.(W,CC,N), E.(N,CC,S,SE), CC.(W,SW,S,N), S.SE, S.SW, SW.W /
     RN(N)   RESERVOIRS /  NW, E  /
     DN(N)   DEMAND NODES; DN(N) = YES; DN(RN) = NO; DISPLAY DN;
     ALIAS   (N,NP);

 TABLE       NODE(N,*)  NODE DATA

                           DEMAND      HEIGHT        X          Y        SUPPLY     WCOST      PCOST
  *                      M**3/SEC  M OVER BASE       M          M       M**3/SEC   RP/M**3    RP/M**4
             NW                         6.50       1200       3600       2.500      0.20       1.02
             E                          3.25       4000       2200       6.000      0.17       1.02
             CC            1.212        3.02       2000       2300
             W             0.452        5.16        750       2400
             SW            0.245        4.20        900       1200
             S             0.652        1.50       2000       1000
             SE            0.252        0.00       4000        900
             N             0.456        6.30       3700       3500

 PARAMETER   DIST(N,N)  DISTANCE BETWEEN NODES (M);
          DIST(N,NP)$A(N,NP) = SQRT( SQR( NODE(N,"X")-NODE(NP,"X") ) + SQR( NODE(N,"Y")-NODE(NP,"Y") ) );
             DISPLAY DIST;

 SCALAR      DPOW        POWER ON DIAMETER IN PRESSURE LOSS EQUATION      / 5.33    /
             QPOW        POWER ON FLOW IN PRESSURE LOSS EQUATION          / 2.00    /
             DMIN        MINIMUM DIAMETER OF PIPE                         / 0.15    /
             DMAX        MAXIMUM DIAMETER OF PIPE                         / 2.00    /
             HLOSS       CONSTANT IN THE PRESSURE LOSS EQUATION           / 1.03E-3/
             DPRC        SCALE FACTOR IN THE INVESTMENT COST EQUATION     / 6.90E-2/
             CPOW        POWER ON DIAMETER IN THE COST EQUATION           / 1.29    /
             R           INTEREST RATE                                    / 0.10    /
             DAVG        AVERAGE DIAMETER (GEOMETRIC MEAN)
             RR          RATIO OF DEMAND TO SUPPLY;

             DAVG = SQRT(DMIN*DMAX);
             RR   = SUM(DN,NODE(DN,"DEMAND")) / SUM(RN,NODE(RN,"SUPPLY"));
```

```

VARIABLES  Q(N,N)       FLOW ON EACH ARC - SIGNED          (M**3 PER SEC)
           D(N,N)       PIPE DIAMETER FOR EACH ARC         (M)
           H(N)         PRESSURE AT EACH NODE              (M)
           S(N)         SUPPLY AT RESERVOIR NODES          (M**3 PER SEC)
           PCOST        ANNUAL RECURRENT PUMP COSTS        (MILL RP)
           DCOST        INVESTMENT COSTS FOR PIPES         (MILL RP)
           WCOST        ANNUAL RECURRENT WATER COSTS       (MILL RP)
           COST         TOTAL DISCOUNTED COSTS             (MILL RP)

EQUATIONS  CONT(N)      FLOW CONSERVATION EQUATION AT EACH NODE
           LOSS(N,N)    PRESSURE LOSS ON EACH ARC
           PEQ          PUMP COST EQUATION
           DEQ          INVESTMENT COST EQUATION
           WEQ          WATER COST EQUATION
           OBJ          OBJECTIVE FUNCTION;

CONT(N)..  SUM( NP , Q(NP,N)$A(NP,N) - Q(N,NP)$A(N,NP) ) + S(N)$RN(N) =E= NODE(N,"DEMAND");

LOSS(N,NP)$A(N,NP)..
           H(N) - H(NP) =E= HLOSS * DIST(N,NP) * ABS(Q(N,NP))**(QPOW-1) * Q(N,NP) / D(N,NP)**DPOW;

PEQ..      PCOST =E= SUM( RN , S(RN) * NODE(RN,"PCOST") * ( H(RN)-NODE(RN,"HEIGHT") ) );

DEQ..      DCOST =E= DPRC * SUM( (N,NP)$A(N,NP) , DIST(N,NP) * D(N,NP)**CPOW );

WEQ..      WCOST =E= SUM( RN , S(RN) * NODE(RN,"WCOST") );

OBJ..      COST  =E= ( PCOST + WCOST ) / R + DCOST;
*
*  BOUNDS
*
D.LO(N,NP)$A(N,NP) = DMIN;                D.UP(N,NP)$A(N,NP) = DMAX;
H.LO(RN)           = NODE(RN,"HEIGHT");   H.LO(DN)           = NODE(DN,"HEIGHT") + 7.5 + 5.0 * NODE(DN,"DEMAND");
S.LO(RN)           = 0;                   S.UP(RN)           = NODE(RN,"SUPPLY");
*
*  INITIAL VALUES
*
D.L(N,NP)$A(N,NP)  = DAVG;
H.L(N)             = H.LO(N) + 5.0;
S.L(RN)            = NODE(RN,"SUPPLY")*RR;
*
MODEL NETWORK /ALL/;
SOLVE NETWORK USING DNLP MINIMIZING COST;
DISPLAY Q.L;
```

```
SYMBOL       TYPE    REFERENCES

A            SET     DECLARED       23       REF     2*42      4*75      2*77      2*82      4*90      2*96
ABS          FUNCT        REF       78
CONT         EQU     DECLARED       68  DEFINED        75
COST         VAR     DECLARED       66 IMPL-DEF       101       REF      2*86
CPOW         PARAM   DECLARED       51      REF      2*82
D            VAR     DECLARED       60 IMPL-DEF     2*101 ASSIGNED     2*90        96       REF      2*78      2*82
DAVG         PARAM   DECLARED       53 ASSIGNED        56       REF      2*96
DCOST        VAR     DECLARED       64 IMPL-DEF     2*101       REF      2*82      2*86
DEQ          EQU     DECLARED       71  DEFINED        82
DIST         PARAM   DECLARED       41 ASSIGNED        42       REF        43      2*78      2*82
DMAX         PARAM   DECLARED       48      REF      2*56      2*90
DMIN         PARAM   DECLARED       47      REF      2*56      2*90
DN           SET     DECLARED       25 ASSIGNED      2*25       REF        25        57      2*91  CONTROL        57        91
DPOW         PARAM   DECLARED       45      REF      2*78
DPRC         PARAM   DECLARED       50      REF      2*82
H            VAR     DECLARED       61 IMPL-DEF     3*101 ASSIGNED     2*91        97       REF      4*78      2*80      2*97
HLOSS        PARAM   DECLARED       49      REF      2*78
LOSS         EQU     DECLARED       69  DEFINED        78
N            SET     DECLARED       15      REF      2*23        24        25        26        28      2*41      3*42      2*59      2*60
                           61       62       68      2*69      7*75        77      5*78      3*82      2*90        96        97
                      CONTROL       25       42        75        77        82      2*90        96        97
NETWORK      MODEL   DECLARED      100
NODE         PARAM   DECLARED       28      REF      8*42      4*57      2*75      4*80      2*84      6*91      2*92      2*98
NP           SET     DECLARED       26      REF      3*42      4*75        77      5*78      3*82      2*90        96  CONTROL        42
                           75       77       82      2*90        96
OBJ          EQU     DECLARED       73  DEFINED        86
PCOST        VAR     DECLARED       63 IMPL-DEF     2*101       REF      2*80      2*86
PEQ          EQU     DECLARED       70  DEFINED        80
Q            VAR     DECLARED       59 IMPL-DEF     4*101       REF      4*75      4*78       102
QPOW         PARAM   DECLARED       46      REF      2*78
R            PARAM   DECLARED       52      REF      2*86
RN           SET     DECLARED       24      REF        57      2*75      4*80      2*84        91        92        98  CONTROL        25
                           57       80       84        91      2*92        98
RR           PARAM   DECLARED       54 ASSIGNED        57       REF      2*98
S            VAR     DECLARED       62 IMPL-DEF     3*101 ASSIGNED     2*92        98       REF      2*75      2*80      2*84
SQR          FUNCT        REF     2*42
SQRT         FUNCT        REF       42       56
WCOST        VAR     DECLARED       65 IMPL-DEF     2*101       REF      2*84      2*86
WEQ          EQU     DECLARED       72  DEFINED        84


SETS

A            ARCS (ARBITRARILY DIRECTED)
DN           DEMAND NODES
N            NODES
NP           ALIASED WITH N
RN           RESERVOIRS


PARAMETERS

CPOW         POWER ON DIAMETER IN THE COST EQUATION
DAVG         AVERAGE DIAMETER (GEOMETRIC MEAN)
```

```
PARAMETERS

DIST        DISTANCE BETWEEN NODES (M)
DMAX        MAXIMUM DIAMETER OF PIPE
DMIN        MINIMUM DIAMETER OF PIPE
DPOW        POWER ON DIAMETER IN PRESSURE LOSS EQUATION
DPRC        SCALE FACTOR IN THE INVESTMENT COST EQUATION
HLOSS       CONSTANT IN THE PRESSURE LOSS EQUATION
NODE        NODE DATA
QPOW        POWER ON FLOW IN PRESSURE LOSS EQUATION
R           INTEREST RATE
RR          RATIO OF DEMAND TO SUPPLY


VARIABLES

COST        TOTAL DISCOUNTED COSTS          (MILL RP)
D           PIPE DIAMETER FOR EACH ARC      (M)
DCOST       INVESTMENT COSTS FOR PIPES      (MILL RP)
H           PRESSURE AT EACH NODE           (M)
PCOST       ANNUAL RECURRENT PUMP COSTS     (MILL RP)
Q           FLOW ON EACH ARC - SIGNED       (M**3 PER SEC)
S           SUPPLY AT RESERVOIR NODES       (M**3 PER SEC)
WCOST       ANNUAL RECURRENT WATER COSTS    (MILL RP)


EQUATIONS

CONT        FLOW CONSERVATION EQUATION AT EACH NODE
DEQ         INVESTMENT COST EQUATION
LOSS        PRESSURE LOSS ON EACH ARC
OBJ         OBJECTIVE FUNCTION
PEQ         PUMP COST EQUATION
WEQ         WATER COST EQUATION


MODELS

NETWORK


COMPILATION TIME =        0.551 SECONDS
```