COMPARISON OF GAMS, AMPL, AND MINOS FOR OPTIMIZATION

XUEYU CHEN, KRISHNARAJ S. RAO, JUFANG YU, AND RALPH W. PIKE
Louisiana State University • Baton Rouge,, AL 70803

Optimization of plant operations and process design requires maximizing a profit function subject to a plant model that can involve thousands of constraint equations. The mathematical programming modeling languages of GAMS and AMPL were developed to alleviate many of the difficulties associated with the development and solution of large, complex mathematical programming models like these and to allow direct formulation and solution on a computer. They have problem formulation in a language very similar to the mathematical statement of the optimization problem.

The modeling language GAMS (General Algebraic Modeling System) was developed at the World Bank to facilitate the solution of multi-sectoral economy-wide models[1] where FORTRAN programs had been previously used. The modeling language AMPL (A Modeling Language for Mathematical Programming) was developed at AT&T Bell Laboratories for communication applications.[2] These two languages offer an efficient and effective way to solve mathematical programming problems at the expense of learning another programming language. Both languages have similar construction, and AMPL is interactive and use separate model and data files.

GAMS appeared in 1988, is now in version 2.25, and has a number of linear, mixed integer linear, nonlinear, and mixed integer nonlinear solvers, including MINOS, CONOPT, CPLEX, DICOPT, LAMPS, XA, and OSL, among others.[3] AMPL appeared in 1993 and includes the solvers MINOS, XA, and OSL, with others to become available.[4] Both have mainframe, workstation, and PC versions, and they have student editions that can solve small problems (about 300 constraint equations). The manual is the same for all versions, and licensing fees are comparable.

GAMS has been used to solve chemical engineering optimization problems, and Grossmann[5] has edited a CACHE Design Case Studies Series with a number of typical problems for use in optimization courses. Also, we have used GAMS and AMPL in research and instruction and have found them to be valuable tools that can be used to solve a range of optimization problems. Consequently, we offer here a brief comparison of GAMS, AMPL, and MINOS to assist those who would like to take advantage of this new approach for solving mathematical programming problems.

Prior to GAMS and AMPL, codes like MINOS were used to solve large linear and nonlinear programming problems. MINOS (Modular In-Core, Non-Linear Optimization System) is a widely used nonlinear programming solver that was developed in the System Optimization Laboratory of the Department of Operations Research at Stanford University. It is described as a FORTRAN-based computer system that solves large-scale linear and nonlinear optimization problems.[6] Two files are needed to solve linear programs. One a MPS (IBM-Mathematical Programming System) file, is required for all problems to define the names of all variables and constraints and to specify the bounds and initial values for variables. The other is a SPECS (Specifications) file that sets various run-time parameters.

For nonlinear programming problems, two additional FORTRAN subroutines, FUNOBJ and FUNCON, are required. The nonlinear parts of the objective function are provided in a FORTRAN subroutine FUNOBJ, and the nonlinear constraints are defined by the subroutine FUNCON. The subroutine FUNOBJ calculates values of the nonlinear part of the objective function and as many gradients as possible. The subroutine FUNCON is used to evaluate the nonlinear

Xueyu Chen *is a PhD candidate in chemical engineering at Louisiana State University.*
Krishnaraj S. Rao *received his MS degree from Louisiana State University in computer engineering and is currently with Intel Corporation in Palo Alto, California.*
Jufang Yu *is a PhD candidate in industrial engineering at Louisiana State University.*
Ralph W. Pike *is the Paul M. Horton Professor of Chemical Engineering at Louisiana State University.*

TABLE 1
GAMS Program for Problem P1

```
$TITLE Example Problem
* Define the variables in the optimization problem
VARIABLES X,Y;
POSITIVE VARIABLES X,Y;
*Specify the values of constants in the problem
PARAMETER CT / value.../;
PARAMETER DT / value.../;
PARAMETER A1 / value.../;
PARAMETER A2 / value.../;
PARAMETER A3 / value.../;
* Define the objective function and constraints
EQUATIONS      OBJFUN
               CON1
               CON2;
OBJFUN.. TCOST=E=F[X] + CT*X+DT*Y;
CON1.. f[X] + A1*Y=E=B1;
CON2.. A2*X + A3*Y=E=B2
*Impose the bounds on the variables
X.UP = U;
Y.UP = U;
X.LO = L;
Y.LO = L;
*Specify the equations included by model 'Example"
MODEL Example/all/;
* Give the solve statement
SOLVE Example USING NLP MINIMIZING TCOST;
* Display the optimal solution
DISPLAY X.L, Y.L;
```

TABLE 2

(a.) AMPL Model File for Problem P1

```
# Input the bounds for the variables in the optimization problem
var X>=L, <=U;
var Y>=L, <=U;
# Define the names of the constants in the problem
param CT;
param DT;
param A1
param A2
param A3
# Define the objective function of the problem
minimize obj: F[X] + CT*X + DT*Y;
# Define the constraint equations of the problem
subject to CON1; f[X] + A1*Y = B1
subject to CON2: A2*X + A3*Y = B2;
```

(b.) AMPL Data File for Problem P1

```
# Input the values of the constants in the problem
param CT :=value....;
param DT :=value....;
param A1 :=value....;
param A2 :=value....;
param A3 :=value....;
```

constraints and as many elements of the Jacobian matrix as possible. The current version of MINOS is 5.4, which added a callable subroutine feature to version 5.3.

GAMS 2.25 is described as a high-level language that makes concise algebraic statements of mathematical programming models in a language that is relatively easy to read and write and hence is easy to understand and implement.[1] Further, the advantages of GAMS over FORTRAN solvers like MINOS are described as providing a computer language for compact representation of large and complex models, allowing changes to be made in model specifications simply and safely, having unambiguous statements of algebraic relationships, and permitting model descriptions that are independent of solution algorithms.

A GAMS program is a collection of statements in the GAMS language. These statements consist of the sentences that define data structures, initial values, and data modifications and of equations that provide relationships among the variables. When problems contain matrices and vectors, sets and indices are used to express these statements in a concise form. The program calls on an adapted version of a solver, such as MINOS, that is controlled by a number of default parameters or "options" similar to the SPECS file in MINOS.

AMPL has essentially all of the features of GAMS but is more flexible and interactive. The process and economic models can be input in segments; debugging and running the optimization can be done with the results viewed. In GAMS, a model file has to be edited, and this file is run in a separate step.

In summary, GAMS and AMPL modeling languages act as a bridge between mathematical programming problems and FORTRAN solvers for problem formulation, and they can apply different solvers to an optimization problem. Also, both have a presolve phase that uses bound tightening procedures and variable substitutions to reduce the number of constraints and variables. On the other hand, FORTRAN solvers provide experienced modelers with more flexibility in setting run-time parameters, which is important for large and complicated problems.

GAMS AND AMPL STATEMENTS OF THE OPTIMIZATION PROBLEM

Both linear and nonlinear programming problems can be expressed in the following standard mathematical form used by MINOS:

$$\begin{array}{lll} \text{minimize} & F(x)+c^{T}x+d^{T}y & \text{objective function} \\ \text{subject to} & f(x)+A_1y=b_1 & \text{nonlinear and linear} \\ & A_2x+A_3y=b_2 & \text{equality constraints} \\ & l\le(x,y)\le u & \text{variable bounds} \end{array} \qquad \text{(P1)}$$

where the vectors (c, d, b_1, b_2, l, u) and the matrices (A_1, A_2, and A_3) are constants, where F(x) is a smooth scalar function, and where f(x) is a vector of smooth functions.[6] P1 is a linear programming problem if x is zero. The objective function gives a measure of the profit or cost of the operation of a plant, and the constraint equations represent material and energy balances, rate equations, equilibrium relations, demand for product, availability of raw material, etc.

The GAMS and AMPL statements are given in Tables 1 and 2 for the mathematical programming problem P1 with the parameters and variables as scalars. The AMPL model file is in Table 2a and the data file is in Table 2b. As can be seen in Tables 1 and 2a,b, the modeling language representations are similar to the mathematical statements for problem P1. Both start by defining variables and parameters and then follow with the objective function and constraints. GAMS

has the values of parameters with their definitions, and AMPL has the values of parameters in a data file. These programs are easy to read, and they can be checked by people other than the modeler.

A nonlinear fuel oil allocation optimization problem by Karimi from the CACHE compilation of GAMS models[5] is given in the appendix with the GAMS, AMPL, and MINOS codes and solutions. This is a representative illustration for the comparison of these three methods. In the next section, results are given for comparisons of eleven small standard engineering optimization problems. Copies of the GAMS, AMPL, and MINOS codes for these problems are available by sending an e-mail request to

chepik@lsuvm.sncc.lsu.edu

COMPARISONS OF STANDARD OPTIMIZATION PROBLEMS

A comparison was made among GAMS, AMPL, and MINOS to evaluate their capability of solving eleven standard engineering optimization problems. These included two linear and nine nonlinear programming problems given by Grossmann,[5] Pike,[7] Hock and Schittkowski,[8] and Schittkowski.[9] A brief description of each problem is given in Table 3, and a summary of the optimization results is given in Table 4. The performance of these three programs was evaluated by comparing the number of major and minor iterations, the number of superbasic variables left at the optimum, and the number of function calls.

In a major iteration of the optimization algorithm, the nonlinear constraints are linearized at a point to give a set of linearized constraints. A major iteration is a step between the linearizations of the nonlinear constraints. The minor iterations are steps of the simplex or reduced gradient method that search for the feasible and optimal solution based on these linearized constraints. For linearly constrained problems, only minor iterations take place. For nonlinearly constrained problems, both major and minor iterations are required, and minor iterations take place between the successive linearizations of the nonlinear constraints. The number of major and minor iterations, especially for nonlinear problems, strongly depends on the initial values and bounds on the variables, the expressions for constraint equations, and the run-time parameters.

In the reduced gradient algorithm, the total of n variables are separated into a set of m basic variables, where m is the number of constraints and (n-m) nonbasic or independent variables. The superbasic variables are subset of the nonbasic variables that can profitably be changed.[11] At the first feasible point, all nonbasic variables away from their bounds are chosen as superbasic,

TABLE 3
Description of Standard Optimization Problems

PROBLEM	*DESCRIPTION*
Refinery Scheduling LP 9 variables 4 eq., 8 ineq. constraints	A refinery produced gasoline, heating oil, jet fuel, and lube oil from limited amount of 4 different crudes. The objective was to maximize the profit per week by increasing product sales and reducing the operating and purchase costs of crude (Karimi in [5])
Petroleum Refinery LP 33 variables 21 eq., 16 ineq. constraints	The objective of this simple, yet non-trivial problem was to find the optimum operating conditions for a refinery that maximized profit. It had three process units, each having several input and output streams, and it had four product streams.[7]
Fuel Allocation NLP 8 variables 2 eq., 6 ineq. constraints	A two-boiler turbine-generator, using a combination of fuel oil and blast furnace gas (limited amount) was used to produce power. The objective was to minimize the consumption of fuel oil required to generate a specified amount of power. The fuel requirements were expressed as a quadratic function of the generated power. (Karimi in [5])
Optimization of Sulfur Content NLP 10 variables 5 eq., 2 ineq. constraints	Three streams having different sulfur intents were combined to form two products having specifications on the maximum sulfur content. The objective was to maximize profit subject to linear and bilinear product and quality constraints.[10]
Alkylation Process Optimization NLP 10 variables 7 eq. constraints	A reactor and fractionator system was used with four feeds to produce alkylate. The objective was to maximize a profit function that included the cost of feed and recycle and sale of product. (Biegler in [5])
Chemical Equilibrium I NLP 12 variables 4 eq. constraints	The objective was to find the equilibrium composition of a mixture of ten chemical species by minimizing the Gibbs free energy subject to elemental balance constraints. This was done by varying the composition of the mixture to arrive at the optimal point. (Karimi in [5])
Chemical Equilibrium II NLP 10 variables 3 eq. constraints	The objective was to find the equilibrium composition by minimizing the Gibbs free energy subject to three elemental balances. [8]
Heat Exchanger Network Configuration - NLP 15 variables 13 eq., 16 ineq. constraints	The objective was to identify the minimum cost for a utility network configuration for a specified combination of process stream matches. (Yee and Grossmann in [5])
A Multi-Spindle Autom. Lathe NLP 10 variables 1 eq., 14 ineq. constraints	The optimization of a multi-spindle automatic lathe was to minimize a nonlinear objective function subject to fifteen generalized polynomial constraints. [9]
Optimization of Linear Objective Function & Quad. Constraints - NLP 15 variables 10 ineq. constraints	This optimization problem was to minimize a linear objective function subject to ten quadratic constraints. [9]
Optimization of Nonlinear Objective Function & Quad. Constraints-NLP 7 variables 2 eq., 3 ineq. constraints	This optimization problem was to minimize a general nonlinear objective function subject to two quadratic and three linear constraints. [9]

TABLE 4
Comparison of Solutions for Standard Optimization Problems with MINOS, GAMS and AMPL

Problem	Solver	No. of Iterations Major	No. of Iterations Minor	Superbasic Var at Opt	No of Function Calls	Obj. Function Value
Refinery Scheduling	MINOS	-	4	-	-	$3.4x10^6/wk
LP	GAMS	-	7	-	-	$3.4x10^6/wk
9 variables	AMPL	-	5	-	-	$3.4x10^6/wk
4 eq., 8 ineq. constraints						
Petroleum Refinery	MINOS	-	32	-	-	$702,000
LP	GAMS	-	26	-	-	$702,000
33 variables	AMPL	-	26	-	-	$702,000
21 eq., 16 ineq. constraints						
Fuel Allocation	MINOS	7	15	1	29	4.681 ton/hr
NLP	GAMS	10	33	1	73	4.681 ton/hr
8 variables	AMPL	7	15	1	47	4.681 ton/hr
2 eq., 6 ineq. constraints						
Optimization of Sulfur Content	MINOS	14	24	0	86	-750 units
NLP	GAMS	14	27	0	70	-750 units
10 variables	AMPL	14	24	0	68	-750 units
5 eq., 2 ineq. constraints						
Alkylation Process Optimization	MINOS	14	19	1	76	$1,154.43/day
NLP	GAMS	16	131	1	750	$1,154.43/day
10 variables	AMPL	13	40	1	206	$1,161.34/day
7 eq. constraints						
Chemical Equilibrium I	MINOS	-	26	7	75	-43.38
NLP	GAMS	-	26	7	76	-43.49
12 variables	AMPL	-	26	7	72	-43.49
4 linear eq. constraints						
Chemical Equilibrium II	MINOS	-	39	7	111	-47.76109
NLP	GAMS	-	21	7	45	-47.76109
10 variables	AMPL	-	31	7	90	-47.76109
3 linear eq. constraints						
Heat Exchanger Network	MINOS	6	8	0	180	$56,825.83
Configuration - NLP	GAMS	8	78	0	22	$56,825.83
15 variables	AMPL	19	29	0	172	$56,825.83
13 eq., 16 ineq. constraints						
A Multi-Spindle Autom. Lathe	MINOS	5	24	0	116	-4,430.088
NLP	GAMS	4	8	0	22	-4,430.088
10 variables	AMPL	4	12	1	78	-4,430.005
1 eq., 14 ineq. constraints						
Optimization of Linear Objective	MINOS	12	117	13	292	-1,840.00
Function & Quad. Constraints - NLP	GAMS	12	200	8	339	-1,840.00
15 variables	AMPL	12	119	11	296	-1,840.00
10 ineq. constraints						
Optimization of Nonlinear Objective	MINOS	4	9	3	46	-37.413
Function & Quad. Constraints-NLP	GAMS	11	53	3	196	-37.413
7 variables	AMPL	12	50	3	226	-37.413
2 eq., 3 ineq. constraints						

and a variable will leave the superbasis if it hits a bound or becomes basic. During the iterations, nonbasic variables are allowed to enter the superbasis before the beginning of each line search, provided their reduced gradients are significantly large. The number of superbasic variables left in the solution at the optimal point indicates the number of nonbasic variables whose optimal values are not on the bounds.

The number of function calls is the number of times that subroutines FUNOBJ and FUNCON have been called to evaluate the nonlinear objective function and nonlinear constraints.[6] The number of functions calls to nonlinear objective and constraint equations is a measure of the computational effort required to reach the optimum.[6]

For the two linear programming problems, the values of the optimum obtained by GAMS, AMPL, and MINOS were the same as shown in Table 4. The only difference was in the number of iterations that each took to reach the optimal solution. This difference probably came from the variations of default initial values and bounds on the variables specified by the three programs.

As shown in Table 4, there were differences in the number of iterations, superbasic variables left at the optimum, and function calls for the solutions of the nine nonlinear problems. For six of the nine nonlinear optimization problems, the same optimal solution was located by the three methods without providing starting points. Also, the optimal solutions were sensitive to the starting points of the variables for two of the problems because of the nonlinearities in the objective function and constraints as described below. These two problems proved to be a challenge for the methods, and typical difficulties were encountered in obtaining the solution of nonlinear optimization problems.

For the alkylation process optimi-

zation, the values of the objective function at the optimum were the same for GAMS and MINOS ($1,154.43/day), which was the same as Grossmann's[5] result. But AMPL gave a slightly better optimal value ($1,161.34/day). This optimal solution had been reported by the original author of the problem, Liebman, *et al.*[12] Grossmann claimed the difference between the optimal results from his GAMS solution and Liebman's solution was likely due to different default tolerances in MINOS. Also, we have shown that this problem has multiple optimal solutions, and several local maxima have been found by giving different starting points. In the absence of a specified starting point, MINOS executed the problem by setting the variables to zero or to a bound (if it was specified) that was closest to zero and exited when an optimum was located. Without good starting points for most of the variables, MINOS was unable to reach the final maximum objective value. But GAMS found the optimal solution with only one variable initialized, and AMPL was able to reach the final optimal solution without the initialization of any variable.

The multi-spindle automatic lathe problem minimized a nonlinear objective function subject to ten nonlinear constraints. For this optimization problem, GAMS successfully located the global optimal solution from different starting points, or even without specifying a starting point. MINOS and AMPL could locate the correct global optimal solution only when a starting point close to the global optimal solution was given. Otherwise, some sub-optimal solutions were found. Also, when this problem was solved using GAMS with the CONOPT solver, re-scaling of variables and constraints was required—otherwise the problem could not be solved. When a starting point close to the global optimal solution was specified for the three methods, GAMS and MINOS found the same optimal value (-4,430.088), but AMPL located a slightly higher value (-4,430.005). This illustrates the need for starting points close to the optimum and scaling of variables and constraint equations.

In Table 5, measures of the computation efficiency are given by the total number of iterations, superbasic variables left, and function calls for the eleven problems. MINOS took fewer iterations and function calls than GAMS and AMPL in total and for most problems. This may be significant for large, complicated problems. But creating the MPS file and FORTRAN subroutine for MINOS is time consuming and prone to errors. These drawbacks for MINOS may supplant its advantage. For example, some of these optimization problems were assigned to students for homework in an optimization course. A few students solved the problems using MINOS in the time allotted, while all found optimal solutions by AMPL and GAMS. Also, they reported that GAMS and AMPL were easier to use than MINOS when starting with no experience with these methods.

All of the problems required well-scaled variables and constraint equations. Scaling is performed by multiplying factors to have the variables and constraints close to a magnitude of one.[1] Scaling is key to obtaining optimal solutions for problems with widely varying values of the variables and constraint equations. The users manuals describe procedures for scaling.

TABLE 5
Comparison of the Computation Efficiency for Eleven Optimization Problems with MINOS, GAMS, and AMPL

	Total of major iterations	Total of minor iterations	Total of superbasic variables left	Total function calls
MINOS	62	317	32	1011
GAMS	75	610	27	1593
AMPL	81	377	31	1255

SUMMARY

Programming and solving standard optimization problems showed that GAMS, AMPL, and MINOS are all effective, and they release modelers from programming optimization algorithms. The comparisons showed that optimization problems are relatively easy to program in GAMS and AMPL, and they offer a choice of solvers and have a presolve phase to reduce model size. In addition, AMPL has features of separate model and data files, flexible output, and options to run batch operations. GAMS provides a comprehensive output summary that is very helpful in detecting model errors, and it is interfaced with more solvers than AMPL now. MINOS could be more robust than GAMS and AMPL, but programming is more difficult. In addition, this is an active area for developments; Floudas describes MINOPT,[13] an automated mixed-integer nonliner optimizer. Also, GAMS has been extended to use the APROS technique to connect the NLP and MILP in the decomposition of MINLP (Paules and Floudas in [5]).

REFERENCES

1. Brooke, A., D. Kendrick, and A. Meeraus, *GAMS: A User's Guide*, Release 2.25, The Scientific Press, San Francisco, CA (1992)
2. Fourer, R., D.M. Gay, and B.W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming,* The Scientific Press, San Francisco, CA (1993)
3. Meeraus, A., *General Algebraic Modeling System,* GAMS Development Corp., Washington, DC (1994)
4. Kernigham, B.W., personal communication (1994)
5. Grossmann, I.E., Ed., *Chemical Engineering Optimization Models with GAMS: CACHE Process Design Case Studies Series,* CACHE Corp., Austin, TX (1991)
6. Murtagh, B.A., and M.A. Saunders, *MINOS 5.4 User's Guide,* Technical Report SOL 83-20R, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, CA (1993)
7. Pike, R.W., *Optimization for Engineering Systems,* Van Nostrand Reinhold Company, Inc., New York, NY (1986)
8. Hock, W., and K. Schittkowski, *Test Examples for Nonlinear Programming Codes,* Springer-Verlag, New York, NY

(1981)
9. Schittkowski, K., *More Test Examples for Nonlinear Programming Codes,* Springer-Verlag, New York, NY (1987)
10. Floudas, C.A., and I.E. Grossmann, "Algorithmic Approaches to Process Synthesis: Logic and Global Optimization," *Fourth Int. Conf. on Founds. of Computer-Aided Prog. Design,* CACHE, American Institute of Chemical Engineers, New York, NY (1995)
11. Drud, A., "CONOPT: A GRG Code for Large Sparse Dynamic Nonlinear Optimization Problems," *Math. Programming,* **31**, 153 (1985)
12. Liebman, J., L. Lasdon, L. Schrage, and A. Waren, *Modeling and Optimization with GINO,* Scientific Press, Palo Alto, CA (1984)
13. Floudas, C.A. *Nonlinear and Mixed-Integer Optimization,* Oxford University Press, New York, NY (1995)

APPENDIX

A FUEL ALLOCATION OPTIMIZATION PROBLEM

This is a simple, nonlinear, allocation optimization given in the CACHE compilation of GAMS models by Karimi.[5] The problem statement has a two-boiler, turbine-generator combination producing a minimum power output of 50 MW, as shown in Figure 1 (next page). Fuel oil and blast furnace gas (BFG) are to be used, and 10 fuel units per hour of BFG are available. A minimum amount of fuel oil is to be purchased to produce the required power from the two generators. The amount of fuel used, F, in tons per hour for fuel oil

TABLE A1

(a.) GAMS Code for Fuel Allocation Optimization[5]

```
$TITLE Power Generation via Fuel Oil

* Define index sets
SETS    G Power Generators /gen1*gen2/
        F Fuels/oil,gas/
        K Constants in Fuel Consumption Equations/0*2/;

*Define and Input the Problem Data
TABLE A(G,F,K)  Coefficients in the fuel consumption equations
                    0          1          2
  gen1.oil        1.4609     .15186     .00145
  gen1.gas        1.5742     .16310     .001358
  gen2.oil        0.8008     .20310     .000916
  gen2.gas        0.7266     .22560     .000778;
PARAMETER PMAX(G) Maximum power outputs of generators
     /GEN1 30.0, GEN2 25.0/;
PARAMETER PMIN(G) Minimum power outputs of generators
     /GEN1 18.0, GEN2 14.0/;
SCALAR GASSUP Maximum supply of BFG in units per h
     /10.0/
PREQ  Total power output required in MW
     /50.0/;

*Design optimization variables
VARIABLES       P(G) Total power output of generators in MW
                X(G,F) Power outputs of generators from specific fuels
                Z(F) Total Amounts of fuel purchased
                OILPUR Total amount of fuel oil purchased;
POSITIVE VARIABLES P, X, Z;

* Define Objective Function and Constraints
EQUATIONS       TPOWER Required power must be generated
                PWR(G) Power generated by individual generators
                OILUSE amount of oil purchased to be minimized
                FUELUSE(F) Fuel usage must not exceed purchase;

TPOWER..        SUM(G, P(G))=G=PREQ;
PWR(G)..        P(G)=E=SUM(F, X(G,F));
FUELUSE(F)..    Z(F)=G=SUM((K,G),a(G,F,K)*X(G,F)**(ORD(K)-1));
OILUSE..        OILPUR=E=Z("OIL");

* Impose Bounds and Initialize Optimization Variables
* Upper and lower bounds on P from the operating ranges
  P.UP(G) = PMAX(G);
  P.LO(G) = PMIN(G);
*Upper bound on BFG consumption from GASSUP
  Z.UP("gas") = GASSUP;
* Specify initial values for power outputs
  P.L(G)=.5*(PMAX(G)+PMIN(G));

* Define model and solve
  MODEL FUELOIL/all/;
  SOLVE FUELOIL USING NLP MINIMIZING OILPUR;

DISPLAY X.L, P.L, Z.L, OILPUR.L;
```

(b.) GAMS Solution for Fuel Allocation Optimization

```
MODEL STATISTICS

BLOCKS OF EQUATIONS     4     SINGLE EQUATIONS     6
BLOCKS OF VARIABLES     4     SINGLE VARIABLES     9
NON ZERO ELEMENTS      16     NON LINEAR N-Z       4
DERIVATIVE POOL         5     CONSTANT POOL       15
CODE LENGTH            81

GENERATION TIME      =0.220 SECONDS
EXECUTION TIME       =0.280 SECONDS   VERID MW2-00-051

     S O L V E  S U M M A R Y

   MODEL FUEL OIL      OBJECTIVE OILPUR
   TYPE NLP            DIRECTION MINIMIZE
   SOLVER MINOS5       FROM LINE 54

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      2 LOCALLY OPTIMAL
**** OBJECTIVE VALUE        4.6809

EXIT - OPTIMAL SOLUTION FOUND
MAJOR ITNS, LIMIT       10      200
FUNOBJ, FUNCON CALLS    0       73
SUPERBASICS             1
INTERPRETER USAGE       0.00
NORM RG / NORM PI       2.532E-10

VARIABLE X.L            Power outputs of generators from specific fuels
                  OIL       GAS
   GEN1          10.114    19.886
   GEN2           3.561    16.439
VARIABLE P.L            Total power output of generators in MW
GEN1 30.000,    GEN2 20.000
VARIABLE Z.L            Total Amounts of fuel purchased
OIL 4.681,    GAS 10.000
VARIABLE OILPUR.L = 4.6809 Total amount of fuel oil purchased
```

TABLE A2

(a.) AMPL Model file for Fuel Allocation Optimization

```
set G;
set F;
set K;

param COEFF{G, F, K} >=0;
param PMAX {g in G};
param PMIN {g in G};
param J {k in K};

var P{g in G} >=PMIN[g], ,<=PMAX[g};
var X{g in G, f in F} >=0;
var Z{f in F} >=0;

minimize purch_oil{f in F}: Z["oil"];
subject to TPWR: sum {g in G} P[g]>=50;
subject to PWR {g in G}: sum {f in F} X[g,f]=P[g];
subject to FUELUSE {f in F}: sum {k in K, g in G} COEFF[g, f, k]*X[g, f]**J[k]=Z[f];
subject to BFG {f in F}: Z["gas"] <=10;
```

(b.) AMPL Data file for Fuel Allocation Optimization

```
set G:=gen1 gen2;
set F:=oil, gas;
set K:=0, 1, 2;

param COEFF:=
  [gen1, *, *]:        0        1        2  :=
              oil    1.4609   0.15186  0.001450
              gas    1.5742   0.16310  0.001358
  [gen2, *, *]:        0        1        2  :=
              oil    0.8008   0.20310  0.000916
              gas    0.7266   0.22560  0.000778
param: PMAX  PMIN:=
  gen1       30       18
  gen2       25       14;
param: J:=
  0  0
  1  1
  2  2;
```

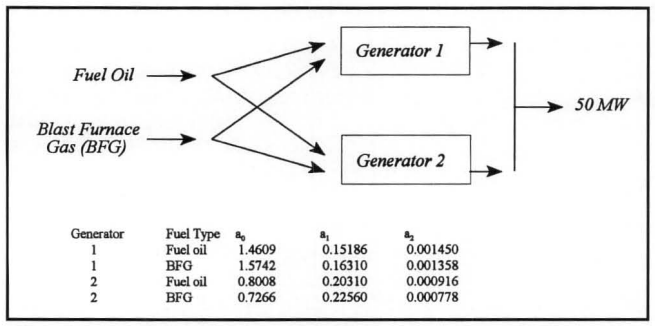
(c.) AMPL Solution for Fuel Allocation Optimization

```
MINOS 5.4:

EXIT-optimal solution found

No. of interations          15      Objective value          4.6808895430E+00
No. of major interations    7       Linear objective         4.6808895430E+00
Penalty parameter           .000100 Nonlinear objective      0.0000000000E+00
No. of calls to funobj      0       No. of calls to funcon   47
No. of superbasics  1       Norm of reduced gradient         1.350E-08
No of basic nonlinears      3       Norm rg / Norm pi        9.610E-09

P[*]    :=gen1   30          gen2          20;
X       :=gen1 gas 19.8857          gen1 oil        10.1143
          gen2 gas 16.4388          gen2 oil        3.56123;
Z[*]:=  gas  10             oil 4.68089;
```



Generator	Fuel Type	a_0	a_1	a_2
1	Fuel oil	1.4609	0.15186	0.001450
1	BFG	1.5742	0.16310	0.001358
2	Fuel oil	0.8008	0.20310	0.000916
2	BFG	0.7266	0.22560	0.000778

Figure 1. *Diagram and parameters for fuel allocation optimization.*[5]

or units per hour for BFG is a quadratic function of the power produced, X, in MW, *i.e.,*

$$F=a_0 + a_1X + a_2X^2$$

where the regression parameters a_0, a_1, and a_2 are listed in Figure 1 for the two fuels and the two generators. Also, the ranges of operation for generators one and two are (18, 30) MW and (14, 25) MW respectively.

The optimal solution will determine the minimum amount of fuel oil to be purchased and its distribution between the two generators. If F_{ij} is the amount of fuel type j (j=1 for fuel oil and j=2 for BFG) used by generator i (i=1,2), then X_{ij} is the corresponding power generated. If Z_1 is the total amount of fuel oil purchased for the two generators, Z_2 is the total usage of BFG for the two generators, and P_i is the power generated by generator I, then the problem can be stated as:

$$\text{Minimize:} \quad Z_1$$

$$\text{Subject to:} \quad \sum_{i=1}^{2} a_{ij0} + a_{ij1}X_{ij} + a_{ij2}X_{ij}^2 \le Z_j \qquad \text{for } j=1,2$$

$$X_{i1} + X_{i2} - P_i = 0 \quad \text{for } i=1,2$$

$$P_1 + P_2 \ge 50$$

$$0 \le Z_2 \le 10$$

$$18 \le P_1 \le 30$$

$$14 \le P_2 \le 25$$

This problem has eight variables and two equality and six inequality constraint equations.

The input files for this problem in GAMS, AMPL, and MINOS are given in Tables A1, A2, and A3. The model statements are similar in GAMS and AMPL, and AMPL has separate model and data files. But the files for MINOS are more complicated, as shown in Table A3a,b, the MINOS MPS and SPC files. The output files are given in Table A1b for GAMS, Table A2c for AMPL, and Table A3d for MINOS, and all three found the same optimal fuel allocation.

TABLE A3

(a.) MINOS MPS File for Fuel Allocation Optimization

```
NAME      FUELOIL
ROWS
 L OIL_AMT
 L GAS_AMT
 E GENT1
 E GENT2
 G TPOWER
 N PUR_OIL
COLUMNS
      X11       GENT1       1.0
      X12       GENT1       1.0
      X21       GENT 2      1.0
      X22       GENT2       1.0
      Z1        PUR_OIL     1.0
      Z2
      P1        GENT1      -1.0        TPOWER     1.0
      P2        GENT2      -1.0        TPOWER     1.0
RHS
      DEMAND TPOWER 50.0
 UP BOUND01      Z2          10.0
 UP BOUND01      P1          30.0
 LO BOUND01      P1          18.0
 UP BOUND01      P2          25.0
 LO BOUND01      P2          14.0
 FR INITIAL      P1          24.0
 FR INITIAL      P2          19.5
ENDATA
```

(b.) MINOS SPC (Specifications) File for Fuel Allocation Optimization

```
BEGIN FUEL OIL  (NLP problem)
*
* To Minimize the Consumption of Fuel Oil for Fuel Oil Allocation
*
 Problem Number    11
 Minimize
 Rows              20
 Columns           30
 Elements          50

 MPS file          10

 Print level       1  *OK for small problems
 Print frequency   1
 Summary frequency 1

 Nonlinear constraints      2
 Nonlinear Jacobian Var     6
 Nonlinear Objective Var    0
 Scale Option               2
END FUELOIL PROBLEM
```

(c.) Funcon Subroutines for Fuel Allocation Optimization

```
PROGRAM          MINOS
IMPLICIT         DOUBLE PRECISION (A-H, O-Z)
PARAMETER        (NWCORE=30000)
DOUBLE PRECISION Z(30000)
CALL MINOS1(Z,NWCORE)
END
********************
SUBROUTINE TCON (MODE, M, N, NJAC, X, F, G, NSTATE, NPROB, Z, NWCORE)
IMPLICIT         DOUBLE PRECISION (A-H, O-Z)
DOUBLE PRECISION X(N), F(M), G(M,N), Z(NWCORE)
COMMON /M1FILE/IREAD, IPRINT, ISUMM
COMMON /M8DIFF/DIFINT(2),GDUMMY,LDERIV,LVLDIF,KNOWNG(2)
F(1)=1.4609  +  (0.15186*X(1)) +  (0.001450*(X(1)**2))
+    + 0.8008 + (0.20310*X(3)) + (0.000916*(X(3)**2)) - X(5)
F(2) = 1.5742 + (0.16310*X(2)) + (0.001358*(X(2)**2))
+    + 0.7266 + (0.22560*X(4)) + (0.000778*(X(4)**2)) - X(6)
G(1,1) = 0.15186 + (2.0*(0.001450)*X(1))
G(1,3) = 0.20310 + (2.0*(0.000916)*X(3))
G(1,5) = -1.0
G(2,2) = 0.16310 + (2.0*(0.001358)*X(2))
G(2,4) = 0.22560 + (2.0*(0.000778)*X(4))
G(2,6) = -1.0
RETURN
END
```

(d.) MINOS Solution for Fuel Allocation Optimization

EXIT - optimal solution found

FUELOIL

No. of iterations	15	Objection value	4.6808896266E+00
No of major interations	7	Linear objective	4.6808896266E+00
Penalty parameter	.00100	Nonlinear objective	0.0000000000E+00
No. of calls to funobj	0	No. of calls to funcon	29
No. of superbasics	1	Norm of reduced gradient	9.160E-07
No. of basic nonlinears	4	Norm rg / Norm pi	9.176E-08
No. of degenerate steps	0	Percentage	.00
Norm of x (scaled)	3.148E+00	Norm of pi (scaled)	9.983E+00

COLUMN	STATE	ACTIVITY	OBJ GRADIENT	LOWER LIMIT	UPPER LIMIT	REDUCED GRADNT
X11	BS	10.11428	.00000	.00000	NONE	.00000
X12	BS	19.88572	.00000	.00000	NONE	.00000
X21	SBS	3.56123	.00000	.00000	NONE	.00000
X22	BS	16.43877	.00000	.00000	NONE	.00000
Z1	BS	4.68089	1.00000	.00000	NONE	.00000
Z2	UL	10.00000	.00000	.00000	10.00000	-.83456
P1	UL	30.00000	.00000	18.00000	30.00000	-.02843
P2	BS	20.00000	.00000	14.00000	25.00000	.00000