KTH Mathematics

Introduction to GAMS

Ulf Brännlund

What is GAMS?

GAMS: General Algebraic Modeling System

- High-level optimization modeling language.
- Makes interpretation from model to algorithm.

Advantages of GAMS

Advantages of GAMS:

- Efficient.
  - Create a set of equations by one statement.
  - Insert data only once.
  - Create prototypes fast.
  - State-of-the-art optimization software.
- Utilizes structure.
- Self-documenting.
- Algebraic representation.
- Large model library.
- Relatively easy to learn.

Other alternatives

Examples of other modeling languages:

- ILOG Studio
- AMPL
- MPL
- LINGO

Examples of other optimization tools:

- Your own C or Fortran code.
- Optimization packages, e.g., CPLEX, MPSX, OSL.
- Interactive optimization programs, e.g., LINDO, LP88.
- Matlab + Optimization Toolbox.
- Excel.

Example

Assume that your model contains the following constraint:

$$\sum_i x_{ij} \geq b_j \quad \text{for all } j.$$

In GAMS you write

```
demand(j)  .. sum(i,x(i,j)) =g= b(j);
```

Note:

- The format is general, in an algebraic notation.
- Not specific for the problem instance.
- The statements are independent of the solver.
- GAMS offers data manipulation of the solution.

A simple transportation problem

- Sets:
  - i factories
  - j customers
- Given data:
  - a_i supply at factory i
  - b_j demand of customer j
  - c_{ij} transportation cost per unit from factory i to customer j
- Decision variables:
  - x_{ij} amount transported from factory i to customer j

A simple transportation problem, cont.

- Objective function
  - Transportation cost: $\sum_{i,j} c_{ij} x_{ij}$
- Constraints
  - Supply: $\sum_j x_{ij} \leq a_i$ for all i.
  - Demand: $\sum_i x_{ij} \geq b_j$ for all j.
  - Nonnegativity: $x_{ij} \geq 0$ for all i, j.

Example of GAMS output

Generating the instance of the demand constraints.

```
DEMAND (J)      ..          SUM (I, X (I, J) )     =G=    B(J);

DEMAND(NEW-YORK)   .. X(SEATTLE,NEW-YORK)   + X(SAN-DIEGO, NEW-YORK) =G= 325
DEMAND (CHICAGO)   .. X (SEATTLE, CHICAGO)   + X(SAN-DIEGO,CHICAGO)   =G= 300;
DEMAND (TOPEKA)    .. X (SEATTLE, TOPEKA)   + X(SAN-DIEGO,TOPEKA) =G= 275;
```

Example of GAMS output, cont.

```
DISPLAY   X.L.    X.M;

                OPTIMAL SOLUTION (IN CASES)
                NEW-YORK        CHICAGO         TOPEKA
SEATTLE          50.000         300.000
SAN-DIEGO       275.000                         275.000

                MARGINAL COSTS (IN $K/CASE)
                 CHICAGO       TOPEKA
SEATTLE                        0.036
SAN-DIEGO        0.009
```

Comments on the transportation problem

`SETS`

- Assignments within `/ /`
- Hyphens (no spaces)
- "Specials":
  - `SET M Machines /MACH01 * MACH24 /;`
    Defines machine MACH01, MACH02, . . . , MACH24.
  - `ALIAS(T,TP);`
    `TP` is another name for the index set `T`.

Comments on the transportation problem, cont.

`PARAMETER, TABLE`

- Lists delimited by `/ /`, elements delimited by comma or return.
- Zero is default.
- Domain check: GAMS' compiler does not accept `"SEATLE"`
- `SCALAR F`

Comments on the transportation problem, cont.

Assignment

```
PARAMETER   C(I,J)  transportation cost

            C(I,J) = F * D(I,J) / 1000;
```

- "=" is the assignment operator.
- `F` and `D(I,J)` muste have been assigned first.
- `C(J,I)` is not tolerated by the compiler.
- There exist many mathematical functions.
- The right hand side of "=" must be computable for GAMS.
- Use assignments instead of "constraints" when possible.

Comments on the transportation problem, cont.

Variables

- Name, domain and comments in the first sentence.
- Type of variable assigned separately or directly.
  - Domain in type assignment.
  - Free variable is default.
- Types are `FREE, POSITIVE, NEGATIVE, BINARY, INTEGER, SOS1, SOS2.`

Note! There is no objective function. Use an equation to define a (free) variable that you want to optimize.

Comments on the transportation problem, cont.

Constraints

- `EQUATION` means equality or inequality.
- `=E=` is different from "=" and `EQ`.
- `=L=` instead of "$\leq$" and `LE`.
- There must be at least one variable in each constraint.
- Format:

```
NAME(DOMAIN)  ..   LHS =E= RHS;
```

Comments on the transportation problem, cont.

- `MODEL` means a set of constraints.
- `/ALL/` means all constraints.
- Alternatively one may specify them, i.e.,

```
MODEL TRANSPRT /COST, SUPPLY, DEMAND/;
```

Comments on the transportation problem, cont.

```
                                  LP
SOLVE MODELNAME USING             NLP    MINIMIZING    OBJECTIVE
                                  MIP    MAXIMIZING
                                  RMIP
```

Effect:

- Generates an instance of the model.
- Creates in put to the solver and hands over the control to the solver.
- Puts the solver's output in GAMS' internal database.
- Gives the control back to GAMS.

Some basic rules

- Create GAMS files with a text editor, alternatively GAMS IDE. It is a pure ASCII file.
- No tabs. (OK with Emacs.)
- Do note use Swedish characters å, ä, ö.
- Arbitrary order of sentences, but declare before use.
- The layout is arbitrary, several rows per statement, several statements per row.
- End each statement by a ";".

Some basic rules, cont.

- Not case sensitive.
- Documenting text in three ways:
  1. Within a declaration, less than 80 characters, no “,”, “;”, “/”.
  2. Asterisk, “*”, at beginning of row.
  3.
```
$ONTEXT ($ at beginning of row)
   This is an explation of the model ...
$OFFTEXT
```

Some basic rules, cont.

- Rules for naming parameters, variables, etc.
  - Referred to as identifiers in user's guide and error output.
  - At most 10 characters (letters or digits).
  - First character must be a letter.
  - Do not use reserved words.
- Rules for naming `SETS`.
  - Referred to as labels in user's guide and error output.
  - At most 10 characters (letters or digits).
  - First character may be a letter or a digit.
  - Do not use reserved words.
- Declaration and assignment separately or jointly.

GAMS' internal database

- GAMS has the following four attributes for variables and equations.
  - `.LO` Lower bound.
  - `.UP` Upper bound.
  - `.L` "Level" (Primal value).
  - `.M` "Marginal" (Dual value).
- These values may be read and written at any time.
- The solver reads all attributes, but only writes in `.L` and `.M`.
- The format is: `IDENTIFIER.ATTRIBUTE(DOMAIN)`

GAMS' internal database, cont.

Example

- Assigning attributes:

```
X.UP(I,J) = CAPACITY(I,J);
X.LO(I,J) = 10;
X.FX("SEATTLE", "NEW-YORK") = 180;
```

`.FX` means fixing, i.e., setting `.LO = .UP`.

- Setting inital values:

```
QUANTITY.L(K) = 0.5 * EOQ(K);
```

- Reading attributes:

```
DISPLAY X.L, X.M;
REPORT(I,"CASES") = SUM(J, X.L(I,J));
```

GAMS output

All output given on a file (filename.lst).

- In case of error
  - Program listing
  - Error messages
  - Cross references
  - List of symbols
- No error
  - Program listing
  - Cross references
  - List of symbols
  - List of equations
  - List of variables
  - Model statistics
  - Status report
  - Solution report

Dollar commands

Dollar commands controls GAMS' output.

The "$" sign must be in the first column.

Example:

- `$TITLE A transportation model` Title on top of each page.
- `$OFFUPPER` Output as upper case and lower case.
- `$OFFSYMLIST OFFSYMXREF` Elminates sybol listing and cross reference listing.
- `$ONTEXT` and `OFFTEXT` GAMS ignores all commands between these statements.

Error messages

- Search for “****”.
- Accept error messages.
- Concentrate on one error mesage at a time.

Advanced features

Undesirable domain control for tables:

```
SETS    I car types /VOLVO, BMW, SAAB /
        J years /Y96*Y99/
TABLE DATA(I,J,*)
        Y96."HAVE"    Y96."NEED"   Y97."HAVE" Y97."NEED"
VOLVO    1                4            4          6
BMW      2                1            6          0
SAAB     3                5            3          4     ;
```

Advanced features

Undesirable domain control for variables.

```
SOLVE TRANSPRT USING LP MINIMIZING Z

PARAMETER REPORT(I,*) optimal produktion per fabrik
REPORT(I,"CASES") = SUM(J, X.L(I,J));
REPORT(I,"SHIP-COST") = SUM(J, C(I,J)*X.L(I,J));
REPORT(I,"SHIPMENTS") = SUM(J$X.L(I,J), 1);

DISPLAY REPORT;
```

Lags and leads

Lags and leads may be constructed via the funktions `ORD(T)` and `CARD(T)`.

Example:

```
SET T time periods /SPRING, SUMMER, FALL, WINTER /;
PARAMETERS TEST1(T), TEST2(T), TEST3(T), TEST4(T), REPORT(T,*)
TEST1(T) = ORD(T);
TEST2(T) = TEST1(T+1);
TEST3(T) = TEST1(T) + 1;
TEST4(T) = TEST1(T++1);
REPORT(T, "TEST1") = TEST1(T);
REPORT(T, "TEST2") = TEST2(T)
REPORT(T, "TEST3") = TEST3(T);
REPORT(T, "TEST4") = TEST4(T);
```

NB! `ORD("FALL")` = 3, and `CARD(T)` = 4.

Lags and leads, cont.

Here is the result:

	TEST1	TEST2(T)	TEST3(T)	TEST4(T)
SPRING	1	2	2	2
SUMMER	2	3	3	3
FALL	3	4	4	4
WINTER	4		5	1

`T-1` and `T--1` are analogus.

Dynamic modeling

```
SET T weeks /W1*W4/;
PARAMETER D(T) /W1 10, W2 12, W3 14, W4 16/;
POSITIVE variables
     X(T) produced quantity
     I(T) storage;
EQUATIONS
     STOREBAL(T) storage balance equations;

STOREBAL(T) ..  I(T-1) + X(T) =E= D(T) + I(T);
```

NB! GAMS ignores indices outside their domains.

Dynamic modeling, cont.

GAMS generates the following equations:

```
--- STOREBAL      =E= storage balance equations
STOREBAL(W1) ..              + X(W1) - I(W1) =E= 10;
STOREBAL(W2) ..      I(W1) + X(W2) - I(W2) =E= 12;
STOREBAL(W3) ..      I(W2) + X(W3) - I(W3) =E= 14;
STOREBAL(W4) ..      I(W3) + X(W4) - I(W4) =E= 16;
```

Special cases "at the ends" are handled separately.

The dollar operator

The dollar operator is helpful in handling exception in summations or definitions of equations etc.

Example in assignment:

```
REPORT(I,"SHIPMENTS") = SUM(J$X.L(I,J), 1);
```

Example in definition of equation:

```
DEMAND(J) .. SUM(I, X(I,J)$(D(I,J) LT DMAX)) =G= B(J);
DEMAND(J) .. SUM(I$(D(I,J) LT DMAX), X(I,J)) =G= B(J);
```

The dollar operator, cont.

Rules:

- `$(condition)` means “such that condition is fulfilled”.
  The condition must be computable for GAMS.

Format:

- $(expression1 GT expression2) (GT, GE, LT, LE, EQ, NE)
- $(expression1) is short for $(expression1 NE 0.0)
- $(expression1 AND condition2) (OR, XOR, NOT)

The dollar operator, cont.

In an assignment using "=", there is a difference between dollar sign to the left and to the right.

- To the left: A$(condition) = B;

  Meaning: If the condition is fulfilled, let A = B. Otherwise, do nothing.

- To the right: A = B$(condition);

  Meaning: If the condition is fulfilled, let A = B. Otherwise, let A=0.

The dollar operator, cont.

- End conditions

```
SCALARS
   INITLAB initial experienced labor force /75/
   ENDLAB  ending experience labor force /90/;
EW.FX(T)$(ORD(T) EQ 1) = INITLAB;
EW.FX(T)$(ORD(T) EQ CARD(T)) = ENDLAB;
```

- Abort

```
SCALARS  TOTCAP, TOTDEM;
TOTCAP = SUM(I, A(I));
TOTDEM = SUM(J, B(J));
ABORT$(TOTDEM GT TOTCAP) TOTCAP, TOTDEM,
     "Total demand exceeds total capacity"
```

- Conditional `DISPLAY`

```
DISPLAY $SHOWX X.L;
```

Loop

Sometimes it may be useful to loop over an index. There is a `LOOP` statement for this purpose. It may be particularly useful prior and after a solution.

Example:

```
SET T years /1985 * 1995/;
SCALARS INITBUD first year budget / 100000 /
        GROWTH  budget growth rate /0.05/ ;
PARAMETER BUDGET(T);
BUDGET(T)$(ORD(T) EQ 1) = INITBUD;
LOOP(T,
    BUDGET(T+1) = (1+GROWTH)*BUDGET(T);
);
```

There may be an arbitrary number of statements inside the loop, including SOLVE statements.

Other flow controls

GAMS has the usual flow controls: for, while and if/else. They are seldom used.

- if/else

```
IF(( X.L GT 0 ),
   S = 1;
ELSEIF( X.L LT 0 ),
   S = -1;
ELSE
   S = 0;
);
```

Other flow controls, cont.

- while

```
MODEL DSGN /ALL/;
OPTION SOLPRINT = OFF;
SCALAR T /1/;
WHILE( (T LE 20),
    X.L(J) = UNIFORM( X.LO(J), X.UP(J) );
    SOLVE DSGN USING NLP MINIMIZING COST;
    DISPLAY X.L, COST.L;
    T = T + 1;
);
```

Other flow controls, cont.

- for

```
MODEL DSGN /ALL/;
OPTION SOLPRINT = OFF;
SCALAR T;
FOR( T = 1 TO 20,
    X.L(J) = UNIFORM( X.LO(J), X.UP(J) );
    SOLVE DSGN USING NLP MINIMIZING COST;
    DISPLAY X.L, COST.L;
);
```

A comment on tolerances

For integer programming, there are two optimality tolerances:

- `optca` absolute optimality tolerance
- `optcr` relative optimality tolerance

These are rather “loose” by default.

To obtain best possible tolerance:

```
option optca=0;
option optcr=0;
```

The meaning of solution in nonlinear programming

- Please make interpretation of optimal solution for nonconvex problems.
- If the method uses first-derivatives only, the point is known to (approximately) satisfy the first-order necessary optimality conditions.
- You may want to try with different initial points.

A warning on powers

The operator “**” requires a positive argument.

This may cause rntime interrupt due to a nonnegative variable being slightly negative.

The operator `power` with argument “2” works for negative variables too.