GAMS Introduction
Salvador Pineda
20/11/13
Welcome to the GAMS Home Page!

The General Algebraic Modeling System (GAMS) is a high-level modeling system for mathematical programming maintainable models that can be adapted quickly to new situations.

- An Introduction to GAMS
- Documentation (including FAQ)
- Contributed Documentation
- Presentations, Books, Posters

- Download Current GAMS System
- Download Older GAMS Systems
- Contributed Software

- Courses and Workshops
- Mailing List, Google Group, and Newsletters

- The GAMS World
- Solution Specialists
- Other Sites on the Web
GAMS appearance
GAMS files

*GMS

Input file

GAMS

Output file

*LST
GAMS files

*GMS
Input file
Additional input files (optional)

GAMS
Model
 Solver
 Solution

*LST
Output file
Additional output files (optional)
GAMS file structure

- Definition of sets, parameters, tables and variables
- Definition of objective function
- Definition of equality constraints
- Definition of inequality constraints
- Definition of the model
- Solve statement
- Display solution (optional)
Example

Transport problem
Transport problem example

Assume that a product is to be shipped in the amounts $u_1, \ldots, u_m$, from each of $m$ shipping origins, and received in amounts $v_1, \ldots, v_n$, by each of $n$ shipping destinations. The problem consists of determining the amounts $x_{ij}$, to be shipped from origins $i$ to destinations $j$, to minimize the cost of transportation.
Transport problem example

1. Data:

   \( m \): the number of origins.
   \( n \): the number of destinations.
   \( u_i \): the amount to be shipped from origin \( i \).
   \( v_j \): the amount to be received in destination \( j \).
   \( c_{ij} \): the cost of sending a unit of product from origin \( i \) to destination \( j \).

2. Variables:

   \( x_{ij} \): the amount to be shipped from origin \( i \) to destination \( j \).

   It is assumed that these variables are non-negative, that is

   \[ x_{ij} \geq 0; \quad i = 1, \ldots, m; \quad j = 1, \ldots, n. \]
Transport problem example

3. **Constraints:** The constraints of this problem are:

\[
\sum_{j=1}^{n} x_{ij} = u_i; \ i = 1, \ldots, m, \\
\sum_{i=1}^{m} x_{ij} = v_j; \ j = 1, \ldots, n,
\]

4. **Function to be optimized.** In the transportation problem we are normally interested in minimizing the total cost of transportation (sum of the unit costs times the amounts being shipped), that is, we minimize

\[
Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}.
\]
Minimize

\[ Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}. \]

subject to

\[ \sum_{j=1}^{n} x_{ij} = u_i; \quad \forall i = 1 \ldots m, \]
\[ \sum_{i=1}^{m} x_{ij} = v_j; \quad \forall j = 1 \ldots n, \]
\[ x_{ij} \geq 0; \quad \forall i = 1 \ldots m; \forall j = 1 \ldots n, \]
Transport problem example

1. Data:

   \( m \): the number of origins.
   \( n \): the number of destinations.
   \( u_i \): the amount to be shipped from origin \( i \).
   \( v_j \): the amount to be received in destination \( j \).
   \( c_{ij} \): the cost of sending a unit of product from origin \( i \) to destination \( j \).

where \( m = n = 3 \) and

\[
C = \begin{pmatrix}
1 & 2 & 3 \\
2 & 1 & 2 \\
3 & 2 & 1
\end{pmatrix}, \quad u = \begin{pmatrix}
2 \\
3 \\
4
\end{pmatrix}, \quad \text{and} \quad v = \begin{pmatrix}
5 \\
2
\end{pmatrix}
\]
## GAMS basic commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set(s)</td>
<td>Name the indices and their possible values</td>
</tr>
<tr>
<td>Scalar(s)</td>
<td>Name the scalars and assign them values</td>
</tr>
<tr>
<td>Parameter(s)</td>
<td>Name the vectors and assign them values</td>
</tr>
<tr>
<td>Table(s)</td>
<td>Name arrays and assign them values</td>
</tr>
<tr>
<td>Variable(s)</td>
<td>Declare variables, assign them a type (optional) and give them upper and lower bounds</td>
</tr>
<tr>
<td>Equation(s)</td>
<td>Name the function to be optimized and the constraints</td>
</tr>
<tr>
<td>Model</td>
<td>Name models and the list of associated constraints</td>
</tr>
<tr>
<td>Solve</td>
<td>Tell GAMS the solver to be used and solve the model</td>
</tr>
<tr>
<td>Display</td>
<td>Tell GAMS the elements to be listed in the output report</td>
</tr>
</tbody>
</table>
SETS

Examples:

- `SET j /m1,m2,m3/;
- `SET j /m1*m3/;
- `SET j /1*3/;
- `SET j explanation /1*3/;

Related commands:

- `Alias (j, jj)
- `Card (j)
- `Ord (j)
** First, indices are declared and defined.
** Index I is used to refer to the three origins.
** Index J is used to refer to the three destinations.
** Note how the symbol `*` is used to list sets members in a compact way.

SETS
I  index of shipping origins /I1*I3/
J  index of shipping destinations /J1*J3/;

\[ m: \text{the number of origins.} \]
\[ n: \text{the number of destinations.} \]
\[ m = n = 3 \]
SCALARS

Examples:

- `SCALAR f /0.056/;
- `SCALAR f explanation /0.056/;`
PARAMETERS (vector of data)

Examples:

- PARAMETER a(i) capacity of plant i in tons
  
  /p1 300
  p2 500/;

- PARAMETER c(i,j) transportation cost
  
  /p1.m1 300
  p1.m2 500
  p2.m1 400
  p2.m2 250 /;
TABLES (matrix of data)

Examples:

- **TABLE** \(d(i,j)\) distance in km

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m2</th>
<th>m3</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>2</td>
<td>1.6</td>
<td>1.8</td>
</tr>
<tr>
<td>p2</td>
<td>2.5</td>
<td>1.2</td>
<td>1.4</td>
</tr>
</tbody>
</table>

- **for**((i,j), \(c(i,j)=3\));

- \(c(i,j) = 3\);

- \(c('p1',j) = 3\);

- \(c('p1','m1') = 3\);
Transport problem GAMS

** Vectors of data (U(I) and V(J)) are defined as parameters
** Data are assigned to vector elements

PARAMETERS
U(I) the amount of good to be shipped from origin I
    /I1 2
    I2 3
    I3 4/
V(J) the amount of good to be received in destination J
    /J1 5
    J2 2
    J3 2/;

\[ u_i: \text{ the amount to be shipped from origin } i. \]
\[ v_j: \text{ the amount to be received in destination } j. \]
\[ u = \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix}, \text{ and } v = \begin{pmatrix} 5 \\ 2 \\ 2 \end{pmatrix} \]
Transport problem GAMS

** The $C(I,J)$ data matrix is defined as a table
** Data are assigned to matrix elements

| TABLE $C(I,J)$ cost of sending a unit from origin $I$ to destination $J$ |
|-------------------|-----|-----|
|                   | J1  | J2  | J3  |
| I1                | 1   | 2   | 3   |
| I2                | 2   | 1   | 2   |
| I3                | 3   | 2   | 1   |

$c_{ij}$: the cost of sending a unit of product from origin $i$ to destination $j$.

$$C = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 2 \\ 3 & 2 & 1 \end{pmatrix}$$
VARIABLES

Examples:

- Variable $x(i,j)$ explanation;
- Variable $a$, $b$, $c(i)$;
- Free Variable $h$;
- Positive Variable $h$;

Types:

Free, Binary, Integer, Positive and Negative

Options:

$x\cdot lo(i,j) = 20$  \hspace{1cm}  $x\cdot up(x,'j1') = 50$

$x\cdot fx('i1','j2') = 50$  \hspace{1cm}  $x\cdot l(i,j) = 50$
Transport problem GAMS

** The $C(I,J)$ data matrix is defined as a table
** Data are assigned to matrix elements

** TABLE $C(I,J)$ cost of sending a unit from origin I to destination J

\[
\begin{array}{ccc}
I1 & J1 & J2 & J3 \\
I2 & 1 & 2 & 3 \\
I3 & 2 & 1 & 2 \\
I3 & 3 & 2 & 1 \\
\end{array}
\]

** The optimization variables are declared.
** First, the objective function variable ($z$) is declared.
** Next, the remaining variables with controlling indices are declared.

** VARIABLES

$z$ objective function variable
$x(I,J)$ the amount of product to be shipped from origin I to destination J;
Transport problem GAMS

** The types of variables are given in the following sentence.
** In the transportation problem, all the variables are positive except
** the objective function variable.

```plaintext
POSITIVE VARIABLE x(I,J);
```

\[ x_{ij} \geq 0; \ i = 1, \ldots, m; \ j = 1, \ldots, n. \]
EQUATIONS

Examples:

• Equation Eq1 Texto descriptivo;
  
  Eq1..  a =e= b*5 + 3 ;

• Equation Eq2(i) Texto descriptivo;
  
  Eq1(i)..  a =g= b(i)*5 + 3 ;

Types:

=g=  =l=  =e=
Transport problem GAMS

** The types of variables are given in the following sentence.
** In the transportation problem, all the variables are positive except
** the objective function variable.

POSITIVE VARIABLE x(I,J);

** The objective function equation is declared.
** The remaining six equations are declared, in a compact way,
** as two index-dependent equations.

EQUATIONS
COST  objective function equation
SHIP(I) shipping equation
RECEIVE(J) receiving equation;
** The following sentences formulate the above declared equations.
** All the constraints are equality constraints (=E=).
** The first one defines the objective function equation as a summation.
** The second equation group represents three
** different single equations depending on index I.
** The left-hand-side is a sum in J of the unknowns x(I,J), and
** the right-hand-side is a previously defined vector of data.
** Similarly to SHIP constraints, the RECEIVE constraints are defined.

```
COST ..  
    z  =E=  
    SUM ((I,J), C(I,J)*x(I,J)) ;
SHIP(I) ..  
    SUM (J, x(I,J))  =E=  U(I) ;
RECEIVE(J) ..  
    SUM (I, x(I,J))  =E=  V(J) ;
```

\[
Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}x_{ij}.
\]

\[
\sum_{j=1}^{n} x_{ij} = u_i; \; i = 1, \ldots, m,
\]

\[
\sum_{i=1}^{m} x_{ij} = v_j; \; j = 1, \ldots, n,
\]
Examples:

- `MODEL Transporte /Eq1, Eq2/;`
- `MODEL Transporte /All/;`
- `MODEL Transporte explanation /All/;`
SOLVE

Examples:

- SOLVE Transpornte using nlp maximizing z;
- SOLVE Transpornte using lp minimizing z;

<table>
<thead>
<tr>
<th>Solver name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>lp</td>
<td>linear programming</td>
</tr>
<tr>
<td>nlp</td>
<td>non-linear programming</td>
</tr>
<tr>
<td>dnlp</td>
<td>non-linear programming with discontinuous derivatives</td>
</tr>
<tr>
<td>mip</td>
<td>mixed integer programming</td>
</tr>
<tr>
<td>rmip</td>
<td>relaxed mixed integer programming</td>
</tr>
<tr>
<td>minlp</td>
<td>mixed integer nonlinear programming</td>
</tr>
<tr>
<td>rminlp</td>
<td>relaxed mixed integer nonlinear programming</td>
</tr>
<tr>
<td>mcp</td>
<td>mixed complementary problems</td>
</tr>
<tr>
<td>mpec</td>
<td>mathematical problems with equilibrium constraints</td>
</tr>
<tr>
<td>cns</td>
<td>constrained nonlinear systems</td>
</tr>
</tbody>
</table>
SOLVE

Suffixes:
- .modelstat
- .solvestat
- .resusd

Options:
- OPTION iterlim = 1e8 ;
- OPTION reslim = 1e10 ;
- OPTION optcr = 0.1 ;
- OPTION optca = 0 ;
Transport problem GAMS

** The following sentences formulate the above declared equations.
** All the constraints are equality constraints (=E=).
** The first one defines the objective function equation as a summation.
** The second equation group represents three
** different single equations depending on index I.
** The left-hand-side is a sum in J of the unknowns x(I,J), and
** the right-hand-side is a previously defined vector of data.
** Similarly to SHIP constraints, the RECEIVE constraints are defined.

```
COST ..
     z =E= SUM((I,J), C(I,J)*x(I,J)) ;
SHIP(I) ..
     SUM(J, x(I,J)) =E= U(I) ;
RECEIVE(J) ..
     SUM(I, x(I,J)) =E= V(J) ;
```

** The next sentence names the model and list its constraints.

```
MODEL transport /COST,SHIP,RECEIVE/;
```
Transport problem GAMS

** The following sentences formulate the above declared equations.
** All the constraints are equality constraints (=E=).
** The first one defines the objective function equation as a summation.
** The second equation group represents three
** different single equations depending on index I.
** The left-hand-side is a sum in J of the unknowns x(I,J), and
** the right-hand-side is a previously defined vector of data.
** Similarly to SHIP constraints, the RECEIVE constraints are defined.

\[
\text{COST ..} \quad z = \text{E} = \text{SUM}((I,J), \text{C}(I,J) \cdot x(I,J)) ;
\]

\[
\text{SHIP(I) ..} \quad \text{SUM}(J, x(I,J)) = \text{E} = \text{U}(I) ;
\]

\[
\text{RECEIVE(J) ..} \quad \text{SUM}(I, x(I,J)) = \text{E} = \text{V}(J) ;
\]

** The next sentence names the model and list its constraints.

```
MODEL transport /COST,SHIP,RECEIVE/;
```

** The next sentence directs GAMS to solve the transportation model using
** a linear programming solver lp to minimize the objective function.

```
SOLVE transport USING lp MINIMIZING z;
```
DISPLAY

Examples:

- `DISPLAY c;`
- `DISPLAY Eq2.m, Eq1.m;`
- `DISPLAY x.1;`
- `DISPLAY "Hello!";`
DEBUGGING

1. Write only a small part of the program and run it

2. Fix the errors, starting from the top of the error messages
   - Windows: click on the error

3. Outcomment part of your program to locate the error
   - * <text>
   - $ontext <text> $offtext
Example

Let’s try
.lst file

---- COST  =E=  objective function equation

COST..  z - x(I1,J1) - 2*x(I1,J2) - 3*x(I1,J3) - 2*x(I2,J1) - x(I2,J2)
        - 2*x(I2,J3) - 3*x(I3,J1) - 2*x(I3,J2) - x(I3,J3) =E= 0 ; (LHS = 0)

---- SHIP  =E=  shipping equation

SHIP(I1).. x(I1,J1) + x(I1,J2) + x(I1,J3) =E= 2 ; (LHS = 0, INFES = 2 ****)
SHIP(I2).. x(I2,J1) + x(I2,J2) + x(I2,J3) =E= 3 ; (LHS = 0, INFES = 3 ****)
SHIP(I3).. x(I3,J1) + x(I3,J2) + x(I3,J3) =E= 4 ; (LHS = 0, INFES = 4 ****)

---- RECEIVE  =E=  receiving equation

RECEIVE(J1).. x(I1,J1) + x(I2,J1) + x(I3,J1) =E= 5 ; (LHS = 0, INFES = 5 ****)
RECEIVE(J2).. x(I1,J2) + x(I2,J2) + x(I3,J2) =E= 2 ; (LHS = 0, INFES = 2 ****)
RECEIVE(J3).. x(I1,J3) + x(I2,J3) + x(I3,J3) =E= 2 ; (LHS = 0, INFES = 2 ****)
### .lst file

**MODEL STATISTICS**

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocks of Equations</td>
<td>3</td>
</tr>
<tr>
<td>Blocks of Variables</td>
<td>2</td>
</tr>
<tr>
<td>Non Zero Elements</td>
<td>28</td>
</tr>
<tr>
<td>Single Equations</td>
<td>7</td>
</tr>
<tr>
<td>Single Variables</td>
<td>10</td>
</tr>
</tbody>
</table>
SOLVE SUMMARY

MODEL  transport
TYPE   LP
SOLVER CPLEX

OBJECTIVE z
DIRECTION MINIMIZE
FROM LINE 77

***** SOLVER STATUS 1 Normal Completion
***** MODEL STATUS 1 Optimal
***** OBJECTIVE VALUE 14.0000

RESOURCE USAGE, LIMIT 0.031 1000.000
ITERATION COUNT, LIMIT 5 20000000000

.lst file
### .lst file

--- EQU SHIP  shipping equation

<table>
<thead>
<tr>
<th></th>
<th>LOWER</th>
<th>LEVEL</th>
<th>UPPER</th>
<th>MARGINAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>2.000</td>
<td>2.000</td>
<td>2.000</td>
<td>.</td>
</tr>
<tr>
<td>I2</td>
<td>3.000</td>
<td>3.000</td>
<td>3.000</td>
<td>1.000</td>
</tr>
<tr>
<td>I3</td>
<td>4.000</td>
<td>4.000</td>
<td>4.000</td>
<td>2.000</td>
</tr>
</tbody>
</table>

--- EQU RECEIVE  receiving equation

<table>
<thead>
<tr>
<th></th>
<th>LOWER</th>
<th>LEVEL</th>
<th>UPPER</th>
<th>MARGINAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>5.000</td>
<td>5.000</td>
<td>5.000</td>
<td>1.000</td>
</tr>
<tr>
<td>J2</td>
<td>2.000</td>
<td>2.000</td>
<td>2.000</td>
<td>EPS</td>
</tr>
<tr>
<td>J3</td>
<td>2.000</td>
<td>2.000</td>
<td>2.000</td>
<td>-1.000</td>
</tr>
</tbody>
</table>
.lst file

--- VAR x the amount of product to be shipped from origin I to destination J

<table>
<thead>
<tr>
<th>LOWER</th>
<th>LEVEL</th>
<th>UPPER</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1.J1</td>
<td>.</td>
<td>2.000</td>
</tr>
<tr>
<td>I1.J2</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>I1.J3</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>I2.J1</td>
<td>.</td>
<td>1.000</td>
</tr>
<tr>
<td>I2.J2</td>
<td>.</td>
<td>2.000</td>
</tr>
<tr>
<td>I2.J3</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>I3.J1</td>
<td>.</td>
<td>2.000</td>
</tr>
<tr>
<td>I3.J2</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>I3.J3</td>
<td>.</td>
<td>2.000</td>
</tr>
</tbody>
</table>
## FUNCTIONS

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs($x$)</td>
<td>Absolute value of $x$</td>
</tr>
<tr>
<td>arctan($x$)</td>
<td>Inverse of the tangent function (in radians)</td>
</tr>
<tr>
<td>ceil($x$)</td>
<td>Smallest integer greater or equal to $x$</td>
</tr>
<tr>
<td>cos($x$)</td>
<td>Cosine function ($x$ in radians)</td>
</tr>
<tr>
<td>erf($x$)</td>
<td>Cumulative distribution function of the normal $N(0,1)$ distribution at $x$</td>
</tr>
<tr>
<td>exp($x$)</td>
<td>Exponential function</td>
</tr>
<tr>
<td>floor($x$)</td>
<td>Largest integer less or equal to $x$</td>
</tr>
<tr>
<td>log($x$)</td>
<td>Natural logarithm of $x$</td>
</tr>
<tr>
<td>log10($x$)</td>
<td>Logarithm in base 10 of $x$</td>
</tr>
<tr>
<td>mapval($x$)</td>
<td>Mapping function</td>
</tr>
<tr>
<td>max($x_1, x_2, \ldots$)</td>
<td>Largest value in the list</td>
</tr>
<tr>
<td>min($x_1, x_2, \ldots$)</td>
<td>Smallest value in the list</td>
</tr>
<tr>
<td>mod($x,y$)</td>
<td>Remainder obtained after dividing $x$ by $y$</td>
</tr>
<tr>
<td>normal($x,y$)</td>
<td>Random number from a normal random variable with mean $x$ and standard deviation $y$</td>
</tr>
<tr>
<td>power($x,y$)</td>
<td>Power function $x^y$ (where $y$ must be an integer)</td>
</tr>
<tr>
<td>$x \ast y$</td>
<td>Power function $x^y$ (where $x$ must be positive)</td>
</tr>
<tr>
<td>round($x$)</td>
<td>Round $x$ to the nearest integer</td>
</tr>
<tr>
<td>round($x,y$)</td>
<td>Rounds $x$ to $y$ decimal places</td>
</tr>
<tr>
<td>sign($x$)</td>
<td>Sign of $x$, 1 if positive, -1 if negative, and 0 if null.</td>
</tr>
<tr>
<td>sin($x$)</td>
<td>Sine function (in radians)</td>
</tr>
<tr>
<td>sqrt($x$)</td>
<td>Square of $x$</td>
</tr>
<tr>
<td>sqrt($x$)</td>
<td>Square root of $x$</td>
</tr>
<tr>
<td>trunc($x$)</td>
<td>Is equal to $\text{sign}(x) \ast \text{floor}(\text{abs}(x))$</td>
</tr>
<tr>
<td>uniform($x,y$)</td>
<td>Random number from a uniform distribution $U(x, y)$</td>
</tr>
</tbody>
</table>
FUNCTIONS SUM and PROD

Examples:

- \textbf{EQUATION} Eq2(j);
  \hspace{1cm} \texttt{Eq2(j)}.. \ a =e= \texttt{sum}(i, \ x(i,j))~; \\

- \textbf{EQUATION} Eq3;
  \hspace{1cm} \texttt{Eq3}.. \ a =e= \texttt{prod}((i,j), \ x(i,j))~; \\

- \textbf{EQUATION} Eq4;
  \hspace{1cm} \texttt{Eq4}.. \ a =e= \texttt{sum}(i, \ x(i,i))~;
**FUNCTION $**

**Examples:**

- \( c(i,j) \geq \text{ord}(i) \geq \text{ord}(j) \) = 1;
- **EQUATION** Eq2(j);
  
  \[ \text{Eq2}(j) \geq \text{ord}(j) > 2 \ldots a = e = 1 \];
- \( a = \text{sum}(i \geq \text{ord}(i) > 2), x(i,i) \);  
- \( y=1; a(y < 5) = 3; \)

**Logic functions:**

and, or, not, xor

\(<, \leq, =, >, >, \lt, \leq, \text{eq}, \neq, \geq, \gt\)
FUNCTION $\$

\[
t = 1 \ldots 7 \\
x(1) + x(2) + x(3) \geq 5 \\
x(2) + x(3) + x(4) \geq 5 \\
\ldots \\
x(5) + x(6) + x(7) \geq 5
\]

SET  t /1*7/;
Alias (t,tt);
EQUATION  Eq(t);
Eq(t)$(\text{ord}(t) \leq 5).$  \[
\text{sum}(t$(\quad \quad \quad \quad \quad \quad ),x(t)) = g = 5;
\]
Eq(t)$(\text{ord}(t) \leq 5).$  \[
\text{sum}(tt$((\text{ord}(tt) \geq \text{ord}(t)) \text{and} (\text{ord}(tt) \leq \text{ord}(t)+2)),x(tt)) = g = 5;
\]
Example:

```plaintext
for(itecount= 1 to 3,
Solve transport using lp minimizing z ;
b(j)=b(j)+ord(j);
);
```
FUNCTION WHILE

Example:

while (itecount le 3,
Solve transport using lp minimizing z ;
b(j) = b(j) + ord(j);
itecount = itecount + 1;
);
Options (GAMS)

**OPTION** limrow

The default output also contains a section called the column listing, analogous to the equation listing, which shows the coefficients of three specific variables for each generic variable. This listing would be particularly useful for verifying a GAMS model that was previously implemented in MPS format. To change the default number of specific column printouts per generic variable, the above command can be extended:

```
option limrow = r, limcol = c ;
```

where c is the desired number of columns. (Setting limrow and limcol to 0 is a good way to save paper after your model has been debugged.)

```
--- SHIP =E=  shipping equation

SHIP(I1)..  x(I1,J1) + x(I1,J2) + x(I1,J3) =E=  2 ;
SHIP(I2)..  x(I2,J1) + x(I2,J2) + x(I2,J3) =E=  3 ;
SHIP(I3)..  x(I3,J1) + x(I3,J2) + x(I3,J3) =E=  4 ;

--- RECEIVE =E=  receiving equation

RECEIVE(J1)..  x(I1,J1) + x(I2,J1) + x(I3,J1) =E=  
```

```
--- SHIP =E=  shipping equation

SHIP(I1)..  x(I1,J1) + x(I1,J2) + x(I1,J3) =E=  2 ;
SHIP(I2)..  x(I2,J1) + x(I2,J2) + x(I2,J3) =E=  3 ;
REMAINING ENTRY SKIPPED

--- RECEIVE =E=  receiving equation

RECEIVE(J1)..  x(I1,J1) + x(I2,J1) + x(I3,J1) =E=  
```
Options (GAMS)

```plaintext
OPTION lp=cplex;
```

<table>
<thead>
<tr>
<th>Solver</th>
<th>License</th>
<th>CNS</th>
<th>DNLP</th>
<th>EMP</th>
<th>LP</th>
<th>MCP</th>
<th>MINLP</th>
<th>MIP</th>
<th>MIQCP</th>
<th>MPEC</th>
<th>NLP</th>
<th>QCP</th>
<th>RMINLP</th>
<th>RMIP</th>
<th>RMIQCP</th>
<th>RMPEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPHAECP</td>
<td>Demo</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AMPL</td>
<td>Demo</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ANTIGONE</td>
<td>Demo</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BARON</td>
<td>Demo</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BDMLP</td>
<td>Demo</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BENCH</td>
<td>Demo</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BONMIN</td>
<td>Demo</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BONMINH</td>
<td>Demo</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CBC</td>
<td>Demo</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CONOPT</td>
<td>Demo</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CONVERT</td>
<td>Demo</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>COUENNE</td>
<td>Demo</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CPLEX</td>
<td>Demo</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DE</td>
<td>Demo</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DECOIS</td>
<td>Demo</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Options (CPLEX)

**compuceserver**  address and port of Cplex remote object server
**depind**  dependency checker on/off
**dettilim**  deterministic time limit
**feasopt**  computes a minimum-cost relaxation to make an infeasible model feasible
**feasoptmode**  Mode of FeasOpt
**feaspref**  feasibility preference
**interactive**  allow interactive option setting after a Control-C
**lpmethod**  algorithm to be used for LP problems
**memoryemphasis**  Reduces use of memory

**lpmethod (integer)**

Specifies which LP algorithm to use. If left at the default value (0 for automatic), and a primal-feasible basis is available, primal simplex will be used. If no primal-feasible basis is available, and **threads** is equal to 1, dual simplex will be used. If **threads** is greater than 1 and no primal-feasible basis is available, the concurrent option will be used. Sifting may be useful for problems with many more variables than equations.

The concurrent option runs multiple methods in parallel. The first thread uses dual simplex. The second thread uses barrier. The next thread uses primal simplex. Remaining threads are used by the barrier run. The solution is returned by first method to finish.

*(default = 0)*

0 Automatic
1 Primal Simplex
2 Dual Simplex
3 Network Simplex
4 Barrier
5 Sifting
6 Concurrent
transport.OptFile = 1;

file opt cplex option file /cplex.opt/;
put opt;
put 'lpmethod 1'/;
putclose;
Thanks for your attention