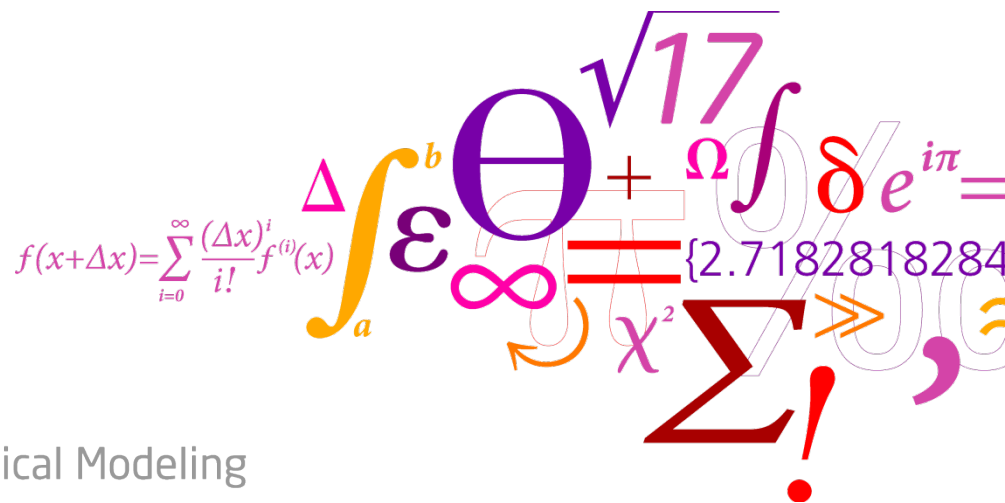


Tools for decision making under uncertainty in electricity markets

GAMS Introduction

Juan Miguel Morales
Salvador Pineda



GAMS download

www.gams.com



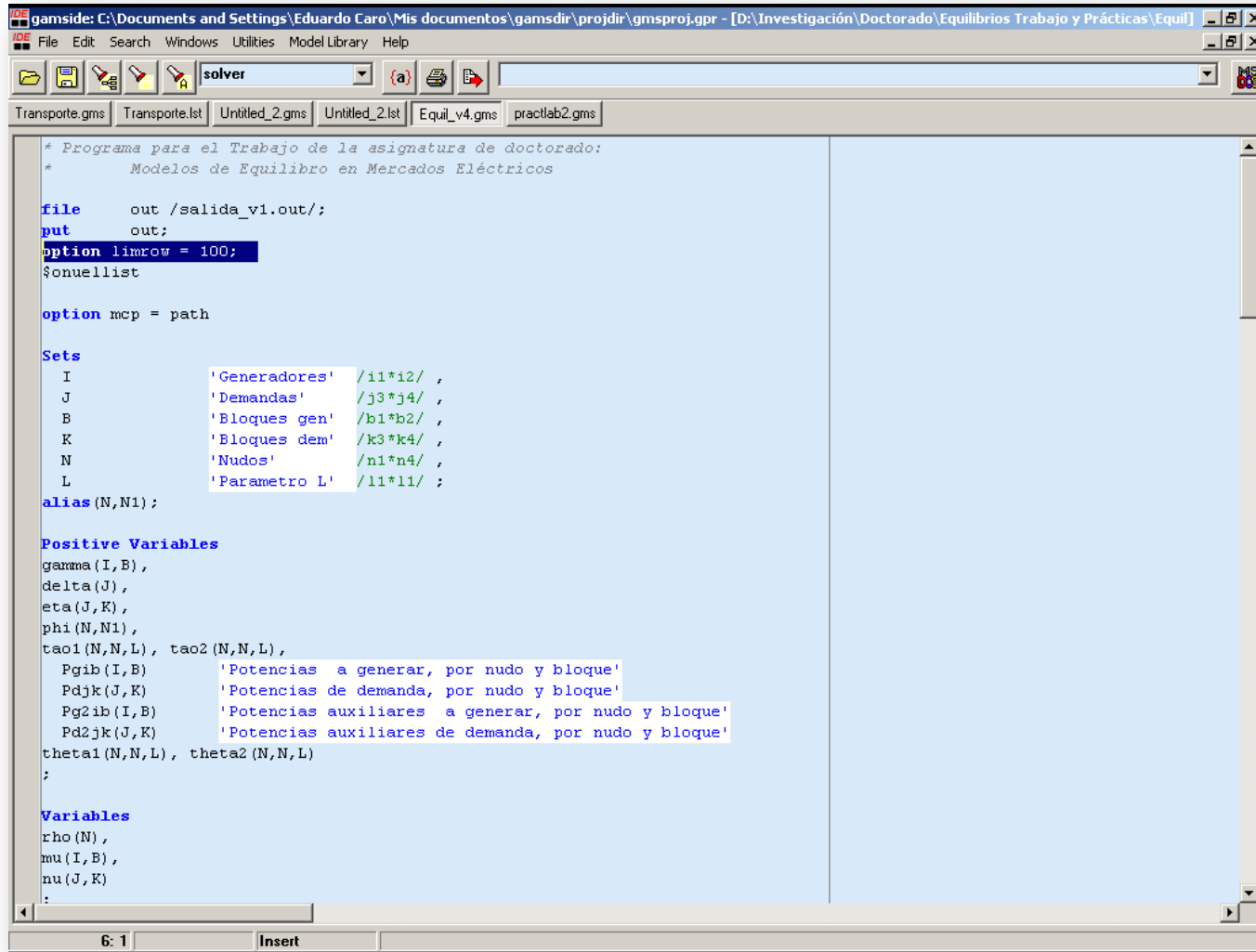
Welcome to the GAMS Home Page!

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



```

* Programa para el Trabajo de la asignatura de doctorado:
* Modelos de Equilibrio en Mercados Eléctricos

file out /salida_v1.out/;
put out;
option limrow = 100;
$onueclist

option mcp = path

Sets
I      'Generadores' /i1*i12/ ,
J      'Demandas'    /j3*j4/ ,
B      'Bloques gen'  /b1*b2/ ,
K      'Bloques dem'  /k3*k4/ ,
N      'Nudos'        /n1*n4/ ,
L      'Parametro L'  /l1*l11/ ;
alias (N,N1);

Positive Variables
gamma(I,B),
delta(J),
eta(J,K),
phi(N,N1),
tao1(N,N,L), tao2(N,N,L),
Pgib(I,B)      'Potencias a generar, por nudo y bloque'
Pdjk(J,K)      'Potencias de demanda, por nudo y bloque'
Pg2ib(I,B)     'Potencias auxiliares a generar, por nudo y bloque'
Pd2jk(J,K)     'Potencias auxiliares de demanda, por nudo y bloque'
theta1(N,N,L), theta2(N,N,L)
;

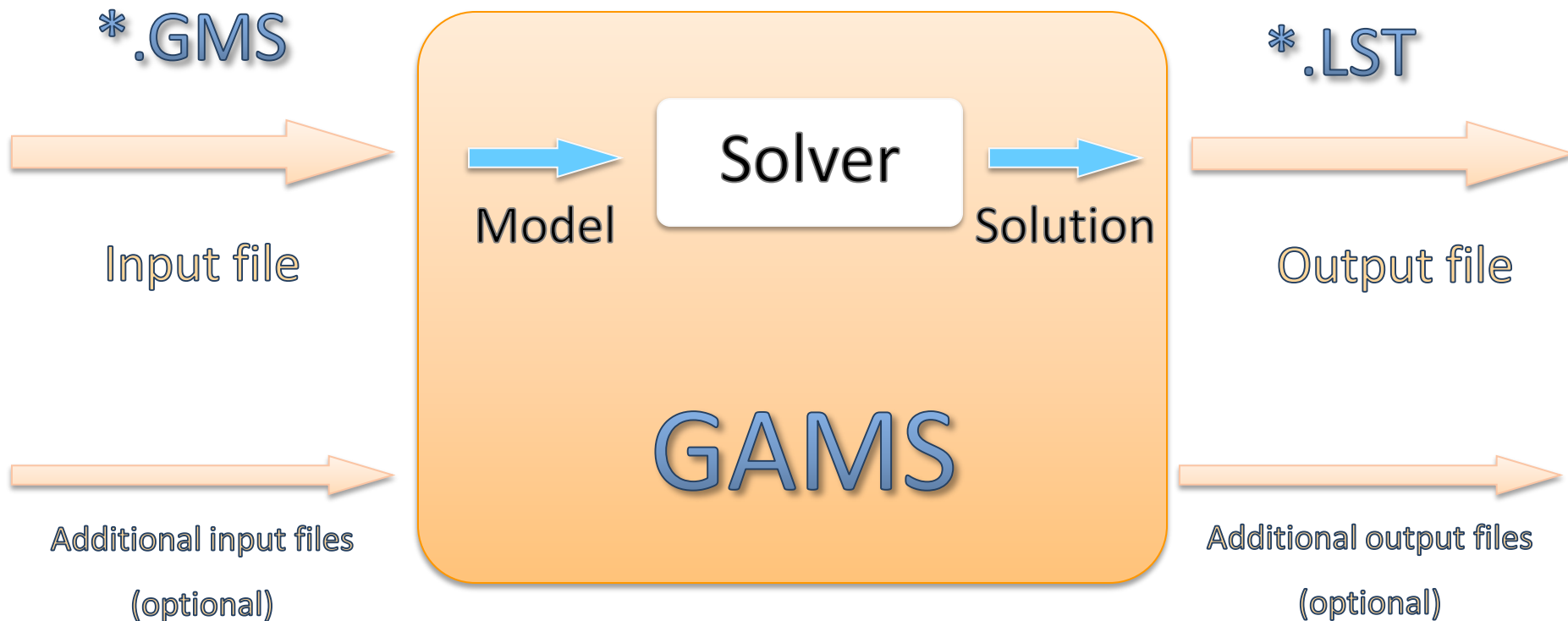
Variables
rho(N),
mu(I,B),
nu(J,K)
;

```

GAMS files



GAMS files



GAMS file structure

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

GAMS basic commands

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

SETS

Examples:

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

Related commands:

- `Alias (N,N1)`
- `Card (N)`
- `Ord (N)`

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`

	m1	m2	m3
p1	2	1.6	1.8
p2	2.5	1.2	1.4;

- `for((i,j), c(i,j)=3);`
- `c(i,j) = 3;`
- `c('p1',j) = 3;`
- `c('p1','m1') = 3;`

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.lo(i,j)=20`

`x.up(x, 'j1') = 50`

`x.fx('i1', 'j2')= 50`

`x.l(i,j)=50`

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=`

MODEL

Examples:

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

SOLVE

Examples:

- `SOLVE Transporte using nlp maximizing z;`
- `SOLVE Transporte using lp minimizing z;`

Solver name	Purpose
lp	linear programming
nlp	non-linear programming
dnlp	non-linear programming with discontinuous derivatives
mip	mixed integer programming
rmip	relaxed mixed integer programming
minlp	mixed integer nonlinear programming
rminlp	relaxed mixed integer nonlinear programming
mcp	mixed complementary problems
mpec	mathematical problems with equilibrium constraints
cns	constrained nonlinear systems

SOLVE

Sufijos:

- `.modelstat`
- `.solvestat`
- `.resusd`

Code	modelstat	solvestat
1	optimal (lp, rmip)	normal completion
2	locally optimal (nlp)	iteration interrupt
3	unbounded (lp, rmip, nlp)	resource interrupt
4	infeasible (lp, rmip)	terminated by solver
5	locally infeasible (nlp)	evaluation error limit
6	intermediate infeasible (nlp)	error
7	intermediate non-optimal (nlp)	-
8	integer solution (mip)	-
9	intermediate non-integer (mip)	-
10	integer infeasible (mip)	-
11	-	-
12	error unknown	-
13	error no solution	-

Opciones:

- `OPTION iterlim = 1e8 ;`
- `OPTION reslim = 1e10 ;`
- `OPTION optcr = 0.1 ;`
- `OPTION optca = 0 ;`

DISPLAY

Examples:

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

FUNCTIONS

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

FUNCTIONS SUM and PROD

Examples:

- `EQUATION Eq2(j);`
`Eq2(j).. a =e= sum(i, x(i,j) );`
- `EQUATION Eq3;`
`Eq3.. a =e= prod((i,j), x(i,j) );`
- `EQUATION Eq4;`
`Eq4.. a =e= sum(i, x(i,i) );`

FUNCTION \$

Examples:

- `c(i,j)$(ord(i) ge ord(j)) = 1;`
- `EQUATION Eq2(j);`
`Eq2(j)$(ord(j) gt 2).. a =e= 1);`
- `a = sum(i$(ord(i) gt 2), x(i,i) );`
- `y=1; a$(y < 5) = 3;`

Logic functions:

`and, or, not, xor`

`<, <=, =, <>, >=, >, lt, le, eq, ne, ge, gt`

FUNCTION FOR

Example:

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

FUNCTION WHILE

Example:

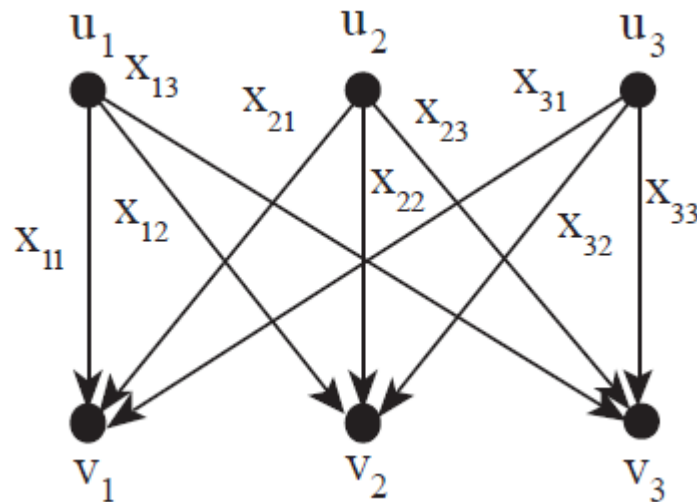
```
while(itecount le 3,  
Solve transport using lp minimizing z ;  
b(j)=b(j)+ord(j);  
itecount = itecount+1;  
);
```

Example

Transport problem

Transport problem example

Assume that a product is to be shipped in the amounts $u_1, \dots, u_m$, from each of m shipping origins, and received in amounts $v_1, \dots, v_n$, 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

$$x_{ij} \geq 0; \quad i = 1, \dots, m; \quad j = 1, \dots, n.$$

Transport problem example

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

$$\begin{aligned}\sum_{j=1}^n x_{ij} &= u_i; \quad i = 1, \dots, m, \\ \sum_{i=1}^m x_{ij} &= v_j; \quad j = 1, \dots, n,\end{aligned}$$

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

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

Transport problem example

Minimize

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

subject to

$$\begin{aligned} \sum_{j=1}^n x_{ij} &= u_i; \quad \forall i = 1 \dots m, \\ \sum_{i=1}^m x_{ij} &= v_j; \quad \forall j = 1 \dots n, \\ x_{ij} &\geq 0; \quad \forall i = 1 \dots m; \forall j = 1 \dots n, \end{aligned}$$

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

$$\mathbf{C} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 2 \\ 3 & 2 & 1 \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix}, \quad \text{and} \quad \mathbf{v} = \begin{pmatrix} 5 \\ 2 \\ 2 \end{pmatrix}$$

Transport problem GAMS

```
** 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 : the number of origins.

n : the number of destinations.

$$m = n = 3$$

Transport problem GAMS

u_i : the amount to be shipped from origin i .

v_j : the amount to be received in destination j .

$$\mathbf{u} = \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix}, \text{ and } \mathbf{v} = \begin{pmatrix} 5 \\ 2 \\ 2 \end{pmatrix}$$

```
** 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/;
```

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 .

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

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;

** 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.
```

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

$$x_{ij} \geq 0; i = 1, \dots, m; j = 1, \dots, n.$$

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;
```

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) ;

```

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

$$\begin{aligned} \sum_{j=1}^n x_{ij} &= u_i; \quad i = 1, \dots, m, \\ \sum_{i=1}^m x_{ij} &= v_j; \quad j = 1, \dots, n, \end{aligned}$$

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.

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/;

**  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;

```

Example

Run the model

.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

BLOCKS OF EQUATIONS	3	SINGLE EQUATIONS	7
BLOCKS OF VARIABLES	2	SINGLE VARIABLES	10
NON ZERO ELEMENTS	28		

.lst file

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

MODEL	transport	OBJECTIVE	z
TYPE	LP	DIRECTION	MINIMIZE
SOLVER	CPLEX	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	2000000000
------------------------	---	------------

.lst file

```

---- EQU SHIP  shipping equation

      LOWER      LEVEL      UPPER      MARGINAL

I1      2.000      2.000      2.000      .
I2      3.000      3.000      3.000      1.000
I3      4.000      4.000      4.000      2.000

---- EQU RECEIVE  receiving equation

      LOWER      LEVEL      UPPER      MARGINAL

J1      5.000      5.000      5.000      1.000
J2      2.000      2.000      2.000      EPS
J3      2.000      2.000      2.000      -1.000

```

.lst file

```

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

      LOWER      LEVEL      UPPER
I1.J1      .      2.000      +INF
I1.J2      .      .      +INF
I1.J3      .      .      +INF
I2.J1      .      1.000      +INF
I2.J2      .      2.000      +INF
I2.J3      .      .      +INF
I3.J1      .      2.000      +INF
I3.J2      .      .      +INF
I3.J3      .      2.000      +INF

```



Thanks for your attention!