Working with GAMS

Part II

Dr. Hussam Alshraideh

Sets

- Simple sets: $S = \{l,k,w\}$ →

```
Set S /l,k,w/
```

- It can also be written as:

```
 Set S first three factors
/l     Labor index
 k     Production index
 w     welfare index/;
```

Multiple names for a set

- Let us consider the following example:

```
Set c /c1,c2/
Table  FoodPrices(c,c)
            c1       c2
       c1    1       5
       c2    5       1;
Parameter cost(c,c);
cost(c,c) = 2.5+10*FoodPrices(c,c);
Display cost;
```

```
What do you expect?  Cost = 12.5     52.5
                            52.5     12.5

But answer will be      Cost  = 12.5      .
                               .        12.5
```

Alias: multiple names of a set

```
Set c /c1,c2/
alias(c,cp);
Table   FoodPrices(c,c)
            c1      c2
       c1    1      5
       c2    5      1;
Parameter cost(c,c);
cost(c,cp) = 2.5+10*FoodPrices(c,cp);
Display cost;
```

Multi-dimensional sets

- GAMS allows up to 20 dimensions

```
set  multidimset(set1name,set2name)
/set1elementname.set2elementname /;
```

e.g

```
Sets
Origins Originating Places  /"New York",Boston/

Destinations  Demand points /Portland,London,Houston/

Linkedbyroad(origins,destinations)
/"NEW York".Portland,
"New York".Houston,
 boston.Portland,
 boston.Houston/;
```

Assigning data for higher dimensions

- The elements in the *n*-tuple are separated by dots (.)

```
sets employee /anderson,hendry,hoffman/
     manager /murphy,smith,morgan/
     department /toy,cosmetics/;
Parameter
Salaries(employee,manager,department)
/anderson.murphy.toy         6000
  hendry.smith.toy         9000
  hoffman.morgan.cosmetics   8000/;
display salaries
```

Tables with more dimensions

```
Sets
i /land,labor/
j /corn,wheat,cotton/
state /al,in/;

Table data(i,j,state) crop
data
                  al       in
land.corn         1        1
labor.corn        6        5
land.wheat        1        1
labor.wheat       4        7
land.cotton       1        1
labor.cotton      8        2;

Display data;
```

```
Sets
i /land,labor/
j /corn,wheat,cotton/
state /al,in/;

parameter data(i,j,state) crop data
/land.corn.al         1
labor.corn.al         6
land.wheat.al         1
labor.wheat.al        4
land.cotton.al        1
labor.cotton.al       8
land.corn.in          1
labor.corn.in         5
land.wheat.in         1
labor.wheat.in        7
land.cotton.in        1
labor.cotton.in       2/;

Display data;
```

Importing data from Excel

- GAMS can read `.gdx` (GAMS Data Exchange) data files.
- Use GDXXRW utility to save Excel data to .gdx format.
- Syntax:

gdxxrw *Inputfile {Outputfile} {Options} [Symbols]*

Inputfile : name of input excel file (.xls, .xlsx)

input=filename.xls   or i=filename.xls

Outputfile: name of data file to be saved (name.gdx)

output=filename.gdx   or o=filename.gdx

Importing data from Excel

- Data Types
  - Par = GAMS_Parameter
  - Equ = GAMS_Equation
  - Var = GAMS_Variable
  - DSet = GAMS_Set
- Data Range
  - Rng = Excel Range
- Dimensions
  - Cdim = Integer (rows where labels are stored)
  - Rdim = Integer (columns where labels are stored)

Importing data from Excel

- Example:
-
```
execute '=GDXXRW input=test.xls output=t1.gdx
par=data rng=Sheet1!C35:F44 Rdim=2 Cdim=1 Dset=i
rng=Sheet1!C35:C44 Rdim=1 Dset=j rng=Sheet1!
D35:D44 Rdim=1 Dset=k rng=Sheet1!E35:F35 Cdim=1';
```

Sets

Loading .gdx data

```
$GDXIN t1.gdx
set i,j,k;
Parameter data(i,j,k);
$LOAD i
$LOAD j
$LOAD k
$LOAD data
$GDXIN
display i,j,k,data;
```

Logical and numerical relationship operators

`lt, <`	Strictly less than
`le, <=`	Less than-or-equal to
`eq, =`	Equal to
`ne, <>`	Not equal to
`ge, >=`	Greater than or equal to
`not`	not
`and`	and
`or`	inclusive or
`xor`	exclusive or

The Dollar Condition

`$(condition)` means ‘such that condition is valid’

- if (cost > 100), then discount = 0.35 can be written as

```
discount$(cost>100) = 0.35
```

- Dollar logical conditions cannot contain variables

- Dollar condition can also be nested

`$(condition1$(condition2))` means `$(condition1 and condition2)`

Dollar on the left

- Consider

```
rho(i)$(sig(i) ne 0) = (1./sig(i)) - 1.;
```

- No assignment is made unless the logical condition is satisfied
- If the parameter on left hand side has not been initialized, then zero will be assigned

Dollar on the Right

- Consider

```
labor = 2$(market > 1.5)
```

- An assignment is always made in this case
- If the logical condition is not satisfied, then the corresponding term will evaluates to 0
- The expression above is equivalent to

if(market > 1.5) then (labor = 2),

else (labor = 0)

Dollar to filter assignments in a set

```
Variable shipped(i,j), total_cost;
Equation costcalc;

costcalc ..
total_cost =e= sum((i,j)$newset(i,j),
            shipcost(i,j)*shipped(i,j));
```

Ord and Card

- Ord returns relative position in a one-dimensional and ordered set

```
set t time periods /2001*2012/
parameter val(t);
val(t) = ord(t);
```

length

- Card returns the number of elements in a set

```
parameter s;
s = card(t);
```

Control structures in GAMs

- If, Else, and Elseif

```
 If (logical condition,
           statements to be executed If true ;
           Elseif logical condition,
            statements executed If this conditional
            is true and the earlier one is false;
            else
            executed when all the previous
                        conditionals were not satisfied;);
```

Control structures in GAMs

```
If(key <= 0,
 data1(i) = -1 ;
 key2=case1;

Elseif((key > 0) and (key < 1)),
 data1(i) = data1(i)**2 ;
 key2=case2;

Elseif((key >= 1) and (key < 2)),
 data1(i) = data1(i)/2 ;
 key2=case3;

else
 data1(i) = data1(i)**3 ;
 key2=case4;
) ;
```

Loop

Syntax

```
Loop((sets_to_vary),
statement or statements to execute
);
```

Example

```
Loop (i,

  mainprice=priceindex(i);
  Solve marketmodel using lp maximizing optim;
  result(i)=optim.l;

) ;
```

While

Syntax

```
While(logical condition,
statement or statements to execute
);
```

Example

```
While (converge = 0 and iter lt lim,

 root=(maxroot+minroot)/2;
 iter=iter+1;
 function_value=a-b*root+c*sqr(root);
 If(abs(function_value) lt tolerance,
         converge=1;
 else
  If(sign(function_value1)=sign(function_value),
     minroot=root;
     function_value1=function_value;
  else
     maxroot=root;
     function_value2=function_value;
  ););él);
```

For

Syntax

```
for (scalar_arg = start_val to(downto) end_val by increment,
 statements;
 );
```

Example

```
for (iter = 1 to iterlimit,

  root=(maxroot+minroot)/2;
  function_value=a-b*root+c*sqr(root);

  If(abs(function_value) lt tolerance,
          iter=iterlim;
  else
   If(sign(function_value1)=sign(function_value),
         minroot=root;
         function_value1=function_value;
   else
         maxroot=root;
         function_value2=function_value;);
   );
  );
```

Repeat

Syntax

```
repeat ( statements to be executed;

   until logical condition is true );
```

Example

```
repeat (

  root=root+inc;
  function_value2= a-b*root+c*sqr(root);

  If((sign(function_value1) ne sign(function_value2)
    and abs(function_value1) gt 0
    and abs(function_value2) gt tolerance),
      maxroot=root;
      signswitch=1
  else
    If(abs(function_value2) gt tolerance,
      function_value1=function_value2;
      minroot=root;));

  until (signswitch>0 or root > maxroot) ;);
```

Sensitivity Analysis with GAMS

- Use the `option` file for `cplex` solver.
- `Cplex.opt`

```
    objrng all
    rhsrng all
```

- Include in main code file

```
    option lp=cplex;
    modelname.optfile=1;
```

Sensitivity Analysis with GAMS

$$
\begin{aligned}
\text{Maximize } z &= 3x_1 + 2x_2 \\
\text{subject to:} & \\
2x_1 + 1x_2 &\leq 100 \\
x_1 + x_3 &\leq 80 \\
x_1 &\leq 40
\end{aligned}
$$

Optimal solution is: $x_1^* = 20$, $x_2^* = 60$, $z^* = 180$.

Include External files

`$Include externalfilename`

- The whole content of the files gets imported
- Include path of the file if it doesn't exist in current working directory
- If extension is not specified, .gms will be added automatically
- To suppress listing of include files
  - `$offinclude` (in main gams file)
  - `$offlisting` (in included file)

Batinclude

- include file with substitution arguments.

```
$batinclude file arg1 arg2 ...
```

Writing to a file

- Use the `PUT` utility in GAMS
- Syntax

```
file fname(s);
put fname;
put item(s);

file fname text / external file name /
```

Writing to a file

```
file factors /factors.dat/, results /results.dat/ ;
put factors ;

put 'Transportation Model Factors'///
    'Freight cost ', f,
    @1#6, 'Plant capacity'/;
loop(i, put @3, i.tl, @15, a(i)/);
put /'Market demand'/;
loop((i,j), put i.tl, @12, j.tl, @24, x.l(i,j):8:4 /);
```

#n   Move cursor position to row n of current page
@n   Move cursor position to column n of current line
/   Move cursor to first column of next line

.ts Displays the text associated with any identifier
.tl   Displays the individual element labels of a set
.te(index) Displays the text associated with an element of a set
.tf   Used to control the display of missing text for set elements

Writing to a file

```
file factors /factors.dat/, results /results.dat/ ;
put factors ;

put ’Transportation Model Factors’///
    ’Freight cost ’, f,
    @1#6, ’Plant capacity’/;
loop(i, put @3, i.tl, @15, a(i)/);
put /’Market demand’/;
loop(j, put @3, j.tl, @15, b(j)/);
put results;
put ’Transportation Model Results’// ;
loop((i,j), put i.tl, @12, j.tl, @24, x.l(i,j):8:4 /);
```