GAMS - Guia rápido de utilização – Parte 1

GAMS (*General Algebraic Modeling System* - http://www.gams.com) é um ambiente para otimização. Os modelos são elaborados na forma de equações algébricas usando uma linguagem de alto nível. O GAMS compila o modelo e faz a interface com o solver selecionado. A extensão do arquivo do modelo é *.GMS, enquanto que a extensão do arquivo de resultados é *.LST. No menu File/Options/Solvers os solvers comerciais são atribuídos às diversas classes de problemas (LP, NLP, MILP...).

O arquivo do modelo, em geral, é organizado nas seguintes seções:

1. Especificação de índices e conjuntos.
2. Declaração e especificação de parâmetros.
3. Declaração das variáveis.
4. Declaração e listagem das equações.
5. Especificação de limites e valores iniciais.
6. Especificação do modelo e chamada do solver.

- Comentários podem ser usados para separar melhor as seções e deixar o arquivo organizado. Comentários são permitidos em linhas exclusivas começando com *.

- Não é necessário especificar qual é a função objetivo dentre as equações do problema. O GAMS maximiza ou minimiza variáveis e não funções. Portanto lance uma equação do tipo $Z = f(\underline{x})$ e otimize a variável Z. A variável a ser otimizada deve ser declarada como contínua e irrestrita (*free variable*).

- Shift+F9 compila o problema, o que é útil para buscar erros de programação, enquanto F9 executa. Quando a lista de erros é exibida, clique duas vezes no primeiro erro em vermelho para apontá-lo. Recomenda-se corrigir os erros sequencialmente, já que erros adiante podem ser simples consequências de erros anteriores.

Exemplo de utilização:

min	$Z = 1{,}5.Y_1 + 2{,}5.Y_2 + 0{,}5.Y_3 + X_1^2 + X_2^2$
sujeito a	$(X_1 - \beta)^2 - X_2 \leq 0$
	$X_1 - X_2 + 4{,}1.(1 - Y_2) \leq 0$
	$X_1 + X_2 + 3.Y_3 - 1{,}3.X_3 = 0$
	$\ln(X_1) - X_3 \leq 0$
	$X_1 \leq 5$
	$X_2 \leq 5$
	$X_1 - 10.Y_1 \leq 0$
	$Y_1 + Y_2 \geq 1$
com	$X_1, X_2, X_3 \geq 0$ reais
	$Y_1, Y_2 = \{0, 1\}$ binárias
	$Y_3 \geq 0$ inteira
	$\beta = 2{,}4$ (parâmetro do modelo)

Formulação no GAMS:

```
*************** Problema de Otimização ***************

****************** Parâmetros ************************

PARAMETER BETA / 2.4 /;

******** Declaração de Variáveis e Equações **********

FREE VARIABLES     Z;
POSITIVE VARIABLES X1, X2, X3;
BINARY VARIABLES   Y1 decisão de produção, Y2 variável lógica;
INTEGER VARIABLES  Y3;

EQUATIONS  OBJ, R1, R2, R3, R4, R5, R6, R7, R8;

******************** Equações ************************

OBJ..     Z =E= 1.5*Y1 +2.5*Y2 +0.5*Y3 +X1**2 +X2**2 ;
R1..      SQR(X1 -BETA) -X2 =L= 0 ;
R2..      X1 -X2 +4.1*(1 -Y2) =L= 0 ;
R3..      X1 +X2 +3*Y3 -1.3*X3 =E= 0 ;
R4..      LOG(X1) - X3 =L= 0 ;
R5..      X1 =L= 5 ;
R6..      X2 =L= 5 ;
R7..      X1 -10*Y1 =L= 0 ;
R8..      Y1 +Y2 =G= 1 ;

************ Limites e Valores Iniciais **************

X1.L = 1 ;
X2.L = 2 ;

******************  Solução  ************************

MODEL Problema / ALL / ;

SOLVE Problema USING MINLP MINIMIZING Z;

*****************************************************
```

- Todas os comandos são encerrados em ponto-e-vírgula, exceto comentários.
- Qualquer nome pode ser atribuído às variáveis e equações na formulação do problema, evitando o uso de caracteres especiais como ‘ç’ ou ‘ã’. Não há diferenciação entre maiúsculas e minúsculas. Nas listas de declaração os nomes devem ser separados por vírgula. É possível inserir um comentário logo após o nome declarado (texto fica azul automaticamente):

  ```
  BINARY VARIABLES   Y1 decisão de produção, Y2 variável lógica;
  ```

- Todas as variáveis do problema devem ser declaradas. Opções:

FREE VARIABLES	variáveis contínuas irrestritas
POSITIVE VARIABLES	variáveis contínuas não-negativas
NEGATIVE VARIABLES	variáveis contínuas não-positivas
BINARY VARIABLES	variáveis binárias tipo 0-1
INTEGRER VARIABLES	variáveis inteiras de 0 a 100

- Parâmetros podem ser declarados usando o comando **PARAMETER** ou **SCALAR**.
- Cada equação é representada algebricamente, iniciando com o seu nome seguido de dois pontos, e encerrando com ponto-e-vírgula. Equações e inequações são representadas através destes operadores:

`=E=`	=	(*equality*)
`=L=`	≤	(*less than or equal to*)
`=G=`	≥	(*greater than or equal to*)

- GAMS possui várias funções matemáticas inclusas (vide manual online), como por exemplo:

`POWER(X,a)`: potência com expoente inteiro. OBS: `X**a` aceita expoente real e base real positiva.
`ABS( )`: valor absoluto.
`COS( ), SIN( ), TAN( )`: coseno, seno, tangente.
`SQR( ), SQRT( )`: elevado ao quadrado e raiz quadrada.
`LOG( ), LOG10( ), EXP( )`: logaritmo natural, logaritmo base 10 e exponencial.
`MIN( ), MAX( )`: mínimo e máximo de uma lista de variáveis.

- Para definir limites inferiores e superiores das variáveis e os chutes iniciais, usam-se os seguintes sufixos:

`.LO`	limite inferior	(*lower bound*)
`.UP`	limite superior	(*upper bound*)
`.L`	chute inicial	(*level*)

Exemplo: $1 \leq X_1 \leq 7$ e iniciar em $X_1 = 3$.

```
X1.LO = 1 ;
X1.UP = 7 ;
X1.L  = 3 ;
```

- Ao final do arquivo, especifica-se um nome para o modelo (no exemplo: `Problema`) e informa-se quais equações compõem o modelo (use `ALL` para todas ou crie uma lista separando por vírgula). Podem ser definidos vários modelos em um mesmo arquivo, usando combinações diferentes de equações.
- A chamada do otimizador é feita da seguinte forma:

`SOLVE` (nome do modelo) `USING` (solver) `MINIMIZING` ou `MAXIMIZING` (variável objetivo)

- Estas são algumas das classes de solvers disponíveis no GAMS:

`LP`	programação linear
`MIP`	programação mista-inteira linear
`RMIP`	programação mista-inteira linear relaxada
`NLP`	programação não-linear com funções suaves
`MINLP`	programação mista-inteira não linear
`RMINLP`	programação mista-inteira não linear relaxada

- Vários solvers comerciais de otimização estão disponíveis no GAMS, tais como CPLEX, CONOPT, DICOPT, BARON, LINDOGLOBAL, ANTIGONE, MINOS, SCIP etc. Ao usar um solver, recomenda-se ler a sua documentação. Testar diferentes solvers para resolver um problema é uma boa estratégia.
- Após compilação e resolução, o resultado é fornecido ao fim de `SOLVE SUMMARY`. Primeiramente são listadas as equações e depois as variáveis. As colunas `LOWER` e `UPPER` indicam os limites viáveis enquanto que a coluna `LEVEL` mostra o valor corrente. Para equações, `LEVEL` representa a parcela constante ao lado direito da expressão e não a folga propriamente dita. Variáveis nulas são indicadas por um ponto ou por `EPS` (*épsilon*, quase zero). A coluna `MARGINAL` traz os multiplicadores referentes às restrições (sinais estão trocados em relação à teoria).

Resultado do GAMS:

```
MODEL STATISTICS

BLOCKS OF EQUATIONS           9     SINGLE EQUATIONS            9
BLOCKS OF VARIABLES           7     SINGLE VARIABLES            7
NON ZERO ELEMENTS            23     NON LINEAR N-Z              4
DERIVATIVE POOL              10     CONSTANT POOL              17
CODE LENGTH                  19     DISCRETE VARIABLES          3

GENERATION TIME      =        0.000 SECONDS      3 MB  24.2.1 r43572 WEX-WEI

EXECUTION TIME       =        0.000 SECONDS      3 MB  24.2.1 r43572 WEX-WEI
GAMS 24.2.1   r43572 Released Dec  9, 2013 WEX-WEI x86_64/MS Windows 07/03/15
09:45:15 Page 5
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
Solution Report     SOLVE Problema Using MINLP From line 33

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

     MODEL   Problema            OBJECTIVE  Z
     TYPE    MINLP               DIRECTION  MINIMIZE
     SOLVER  LINDOGLOBAL         FROM LINE  33

**** SOLVER STATUS     1 Normal Completion
**** MODEL STATUS      1 Optimal
**** OBJECTIVE VALUE                7.2366

 RESOURCE USAGE, LIMIT          0.078      1000.000
 ITERATION COUNT, LIMIT       111    2000000000
 EVALUATION ERRORS             NA             0


                        LOWER     LEVEL     UPPER    MARGINAL

---- EQU OBJ              .         .         .        1.000
---- EQU R1             -INF        .         .       -1.563
---- EQU R2             -INF     -4.100    -4.100    -0.981
---- EQU R3              .         .         .         EPS
---- EQU R4             -INF     -1.716      .         .
---- EQU R5             -INF      1.272     5.000      .
---- EQU R6             -INF      1.272     5.000      .
---- EQU R7             -INF     -8.728      .         .
---- EQU R8             1.000     2.000     +INF       .

                        LOWER     LEVEL     UPPER    MARGINAL

---- VAR Z              -INF      7.237     +INF       .
---- VAR X1               .       1.272     +INF       .
---- VAR X2               .       1.272     +INF       .
---- VAR X3               .       1.957     +INF       .
---- VAR Y1               .       1.000     1.000     1.500
---- VAR Y2               .       1.000     1.000    -1.523
---- VAR Y3               .         .       +INF      0.500


EXECUTION TIME       =        0.016 SECONDS      2 MB  24.2.1 r43572 WEX-WEI

USER: GAMS Development Corporation, Washington, DC   G871201/0000CA-ANY
      Free Demo,  202-342-0180,  sales@gams.com,  www.gams.com   DC0000
```

GAMS - Guia rápido de utilização – Parte 2

A linguagem GAMS (http://www.gams.com) permite a declaração de parâmetros que, diferentemente de variáveis, assumem valores fixos especificados. Outra vantagem da linguagem é o uso de índices, conjuntos, vetores e matrizes, muito úteis para problemas lineares. Estas características são ilustradas a seguir por meio de um exemplo (Rosenthal, GAMS - A User's Guide, 2015).

PROBLEMA: As plantas de Seattle e de San Diego têm capacidades de produção mensal de 350 e 600 lotes de enlatados, respectivamente. A produção deve atender às demandas mensais de New York, Chicago e Topeka, que são de 325, 300 e 275 lotes, respectivamente. Formule um LP para minimizar os custos de distribuição dos enlatados. A tabela abaixo apresenta a distância em milhares de milhas entre plantas e centros de distribuição. O custo de transporte é de $90,00 por lote por mil milhas.

	Mercados		
Plantas	New York	Chicago	Topeka
Seattle	2,5	1,7	1,8
San Diego	2,5	1,8	1,4

MODELAGEM:

Variáveis: z custo total (k$)

x_{ij} quantidade transportada de i para j (lotes)

Parâmetros: a_i capacidade da planta i (lotes)

b_j demanda do mercado j (lotes)

d_{ij} distância de i para j (milhares de milhas)

f custo de transporte ($/lote.milha)

$c_{ij} = f.d_{ij}$ custo de transporte de i para j (k$/lote)

Modelo:

$$\min \quad z = \sum_i \sum_j c_{ij}.x_{ij}$$

s.a.:

$$\sum_j x_{ij} \le a_i \qquad \forall\, i$$

$$\sum_i x_{ij} \ge b_j \qquad \forall\, j$$

$$x_{ij} \in \Re^1$$

Parte 1: declaração dos índices e conjuntos

```
********** Problema de Otimizacao **********
**********  Conjuntos  **********
Sets
  i plantas   / seattle, san-diego / ,
  j mercados  / new-york, chicago, topeka /  ;
```

Parte 2: Declaração e especificação de parâmetros

```
**********  Parâmetros  **********
Parameters
   a(i) capacidade da planta i (lotes)
       / seattle   350
         san-diego 550 / ,
   b(j) demanda do mercado j (lotes)
       / new-york  325
         chicago   300
         topeka    275 / ;

Table d(i,j) distâncias (milhares de milhas)
               new-york  chicago  topeka
    seattle     2.5        1.7      1.8
    san-diego   2.5        1.8      1.4  ;

Scalar f custo de transporte (dolares por lote por milhas) / 0.090 / ;

Parameter c(i,j) custo de transporte (milhares de dolares por lote) ;
  c(i,j) = f * d(i,j) ;
```

- Parâmetros numéricos podem ser especificados como escalar por **`Scalar`**, como vetor por **`Parameter`** ou como matriz por **`Table`**, conforme exemplos acima. Domínios com mais dimensões podem ser declarados como **`Parameter`** ou **`Table`**. Pares elemento-valor em **`Parameter`** podem ser fornecidos em qualquer ordem. Valores não fornecidos são considerados nulos. Valores em branco em **`Table`** são considerados nulos.
- Forma alternativa de fornecer os pares elemento-valor:

```
Parameter a(i) capacidade da planta i (lotes)
        / seattle =  350, san-diego = 550 / ;
```

- Parâmetros podem ser calculados a partir de outros parâmetros ou número usando o sinal de atribuição "=", como mostrado no exemplo acima após a declaração de `c(i,j)`. Analogamente, um valor pode ser atribuído diretamente a um parâmetro já declarado como, por exemplo: `c('seattle','chicago') = 1.7;`

Parte 3: Declaração das variáveis

```
**********  Variáveis  **********
Free Variable
  z       custo total de transporte (milhares de dólares) ;

Positive Variable
  x(i,j)  quantidade transportada (lotes);
```

Parte 4: Declaração e listagem das equações

```
*****************************  Equações  *********************************
Equations
  cost       definição da função objetivo,
  supply(i)  respeitar capacidade planta i,
  demand(j)  satisfazer demanda mercado j ;

cost      ..  z =E= sum((i,j), c(i,j)*x(i,j)) ;
supply(i) ..  sum(j, x(i,j)) =L= a(i) ;
demand(j) ..  sum(i, x(i,j)) =G= b(j) ;
```

- Variáveis e equações distribuídas em domínios devem ser identificadas pelos índices dos conjuntos.

- Somatórios são inseridos nas equações como `sum(,)` e produtórios como `prod(,)` tendo o conjunto de índices como primeiro argumento e a equação como segundo argumento, conforme exemplos.

Parte 5: Especificação do modelo e chamada do solver

```
*****************************  Solução  **********************************
Model problema / all / ;
Solve problema using LP minimizing z ;
```

- Na saída do software, equações e variáveis indexadas são apresentadas de maneira clara. Exemplos:

```
---- EQU demand  satisfazer demanda mercado j
             LOWER     LEVEL     UPPER    MARGINAL
new-york   325.000   325.000     +INF       0.225
chicago    300.000   300.000     +INF       0.153
topeka     275.000   275.000     +INF       0.126
```

```
---- VAR x  quantidade transportada (lotes)
                     LOWER     LEVEL     UPPER    MARGINAL
seattle  .new-york       .      50.000     +INF        .
seattle  .chicago        .     300.000     +INF        .
seattle  .topeka         .         .       +INF      0.036
san-diego.new-york       .     275.000     +INF        .
san-diego.chicago        .         .       +INF      0.009
san-diego.topeka         .     275.000     +INF        .
```

Outras dicas úteis para o uso do GAMS:

- Os elementos dos conjuntos são tratados como caracteres e não como números. Nas equações os elementos são especificados entre aspas simples. Ex: `x('seattle','chicago')` é o elemento $x(1,2)$. Caso o valor da posição do elemento no conjunto precise ser usado em uma equação, adote a função `ord(i)`. Analogamente, o número de elementos em um conjunto é dado por `card(i)`.

- Para criar conjuntos com elementos em sequência numérica use o asterisco. Ex: `/1*3/` equivale a {1, 2, 3}; `/m4*m7/` equivale a {m4, m5, m6, m7}.

- Para evitar convergência prematura em MILPs, recomenda-se alterar o parâmetro de tolerância Optcr (o valor padrão é 0,10):

```
Model problema / all / ;
problema.optcr = 0 ;
```

- Um pseudônimo pode ser atribuído a um conjunto usando **alias**. Isso é útil ao lidar com combinações. Ex: problema do caixeiro viajante (partidas `i` e destinos `ii` referem-se ao mesmo conjunto de cidades).

```
Sets
  i cidades / 1*25 / ,
Alias (i,ii);
Binary variable X(i,ii) viagem de i para ii ;
```

- Alguns problemas requerem o uso de somatório condicional. Exemplo:

$\sum_{i,i\neq j} x_{ij}$ é expresso como `sum(i$(ord(i) ne ord(j)), x(i,j))`

Tem-se a condição de somar $x(i,j)$ em i tal que i seja diferente de j (`ne` = *not equal*)

- Para o caso de somatórios apenas com elementos selecionados, pode-se declarar um subconjunto para simplificar a notação.

Por exemplo, a inequação: $y_{1,3} + y_{1,5} + y_{2,3} + y_{2,5} + y_{4,3} + y_{4,5} + y_{6,3} + y_{6,5} \geq 1$

ficaria

```
Y('1','3') + Y('1','5') + Y('2','3') + Y('2','5') + Y('4','3') + Y('4','5')
+ Y('6','3') + Y('6','5') =G= 1 ;
```

Alternativamente, declara-se um subconjunto de (i,j) no começo do arquivo:

```
SET i / 1*6 /  ;
SET j / 1*6 /  ;
SET subconj(i,j)  / 1.3, 1.5, 2.3, 2.5, 4.3, 4.5, 6.3, 6.5 / ;
```

E usa-se uma equação limpa:

```
sum(subconj(i,j),Y(i,j)) =G= 1
```