# GAMS Notes

# 1. GAMS简介

- GAMS[1]: General Algebraic Modeling System

- GAMS官网中的Documentation选项包含的内容对于学习GAMS的学习者来说是非常有益的，尤其是GAMS官网给出的Model Libraries(模型库)和User's Guide(用户手册)等文件对于学习者非常方，也可下载用户手册PDF版本
    - User's Guide英文版PDF文件除了在官网可以下载之外，还可以直接在GAMS软件中通过 `Ctrl+F` 直接打开
    - User's Guide的中文版可参考魏传江与王浩翻译的《GAMS用户指南》
- 关于GAMS中文网站
    - 数学建模社区
    - 英飞咨询：GAMS软件的官方代理
- 相关中文书籍
    - 潘浩然：《可计算一般均衡建模初级教程》
    - 张欣：《可计算一般均衡模型的基本原理与编程》 第二版
        - 格致出版社官网所提供的本书中CGE模型的GAMS程序链接地址[2]

# 2. GAMS基本结构[3]

## GAMS基础概述

GAMS中使用的术语：

- 索引被称为集合(set)
- 已知数据(外生变量)被称为参数(parameter)
- 决策变量(内生变量)被称为变量(variable)
- 约束和目标函数被称为方程(equation)

针对GAMS的一般性说明：

- 创建GAMS实体包括两个步骤：声明和赋值(定义)
  - 声明指宣布实体存在并给出名称
    - 模型的实体名称必须以字母开头，后面可以跟字母或数字
  - 赋值(定义)指给出实体特定的指或形式
- 使用每一实体(如，集合、参数、变量、方程、模型、求解)首先要进行相应的实体声明(前面括号的内容分别对应的声明语句如下， `set`、`parameter`、`variable`、`equation`、`model`、`solve` )
- 每个语句要用分号结束
- 编译器不区分大小写
- 注释
  - 注释行：以 `*` 开始
  - 注释段落： `$ontext` 和 `$offtext` 之间的内容属于注释
  - 文件文本可以在特定的GAMS语句内插入，作为注释，属于可选项

## 2.1集合

- 集合是GAMS模型的基本模块
- A. 首先用set或sets声明集合，然后给 集合命名 ， 最后给 集合赋值(或者定义) 元素

```
Set
i 'canning plants' / seattle,  san-diego /
j 'markets'        / new-york, chicago, topeka /;
```

  - GAMS使用斜线( / )来描述集合
  - 多字名称如 Bei Jing是不可以的，需要插入连字符号( - )，即写作 Bei-Jing
  - 命名的名字之后可以插入作为解释用的文本，解释文本属于可选项，GAMS编译器不会解释文本内容，但会原样返回

- GAMS中，声明使用复数还是单数，没有差别，GAMS都视为同义
- 在给有序排列的集合元素赋值时，使用星号( * )会很方便，
  如， `/2000 * 2002/ => /2000, 2001, 2002/`
- 集合元素是以字符串保存的，无论集合元素是数字还是文本
- 文件文本的引号可删除

- B. alias语句，用于对先前声明的集合给出另一个名字，例如 `alias(s,ss)`
  - 模型中，当涉及到同一集合内元素的相互作用(多出现在动态集合)或者多维集合时是有用的

# 2.2 数据

- 数据的输入有三种不同的方式
  - 列表赋值
  - 表格赋值
  - 直接赋值

## 2.2.1 列表数据输入(列表赋值)

```
Parameter
  a(i) 'capacity of plant i in cases'
       / seattle    350
         san-diego  600 /

  b(j) 'demand at market j in cases'
       / new-york   325
         chicago    300
         topeka     275 /;
```

- 上面的GAMS语句使用parameter 声明 了两个参数，分别 命名 为a和b，并分别声明( 指定 )它们的域为i和j，之后分别给出了两个参数的文件文本(用于解释， 选择项 )，最后对域i和j的每个元素 赋值
- Note:
  - 可以把上述的两个语句分开写，只需在每个语句之前用parameter声明，然后每个语句末尾添加分号
  - 域元素列表可以用任何方式排列，但是整个列表需要用斜线括住，每队元素-数值必须用逗号分割或以不同的行开始
  - 标量(scalar)是没有域的参数，可以用scalar进行语句声明和赋值列表，只是该列表仅包含一个数值，例如：

    ```
    Scalar f 'freight in dollars per case per thousand miles' / 90 /;
    ```

  - 零是所有参数的缺省值，在元素-数值列表中只需包含非零值
  - GAMS编译器具有范围检测功能，即检测每个列表中的域元素是否是集合的成员

## 2.2.2 表格数据输入(表格(矩阵)赋值)

```
Table d(i,j) 'distance in thousands of miles'
             new-york  chicago  topeka
  seattle         2.5      1.7     1.8
  san-diego       2.5      1.8     1.4;
```

- 上面的GAMS语句使用table 声明 了表格参数，并把参数 命名 为d，之后指定参数的域为在i和j笛卡尔乘积中的有序偶的集合，之后对域元素赋值
- Note：如果表格中是空白输入，则默认为0

## 2.2.3 直接赋值数据输入

```
Parameter c(i,j) 'transport cost in thousands of dollars per case';
c(i,j) = f*d(i,j)/1000;
```

- 第一个语句用于声明参数**c**，指定域(i,j)，提供文件文本
  - 末尾必须有分号
- 第一个语句将参数f和d(i,j)的乘积赋值为c(i,j)
  - 前提是前面的语句中已经给出f和d(i,j)的值或赋值
  - 用单引号括住元素名称，可以给域里的特定元素赋值，如 `c('seattle','new-york')=5`
- 相同的参数可以多次赋值，新赋值语句内容会覆盖先前赋值语句内容，但会相同的参数不可以多次声明
- 赋值语句的右边可以包含多种数学表达式和内部函数

# 2.3 变量

表达gams模型的决策变量(又称为内生变量)必须用variable(s)语句声明，之后每个变量都给定一个名字、一个域(如果适合)、文件文本(可选项)，如：

```
Variable
   x(i,j) 'shipment quantities in cases'  # 为每对(i,j)声明运输变量
   z      'total transportation costs in thousands of dollars'; # z是标量，没有声明域
```

- 最优化的变量必须是一个标量，且必须是任意类型：
  - z是标量，没有声明域，对于GAMS中的最优化模型都必须包含这样的一个变量，作为最优化的值(最大最小数量)
- 每个变量必须赋予一个类型，GAMS允许的常见类型：

| 变量类型 | 变量的允许范围 |
| --- | --- |

| 变量类型 | 变量的允许范围 |
| --- | --- |
| free(任意数，默认) | (-∞,+∞) |
| positive | (0,+∞) |
| negative | (-∞,0) |
| binary | (0 or 1) |
| integer | 0,1,2,...,100(默认) |

- 域在类型赋值中不能重复，域里的所有实体都具有相同的变量类型
- 限制变量为非负的语句

```
Positive Variable x;
```

# 2.4 方程

GAMS建模的功能主要通过建立方程和不等式表现出来，在GAMS中所有的方程和不等式同时建立

## 2.4.1 方程声明和定义

通过关键字 `equation` 进行方程声明，后面紧跟着名字、域和被声明方程或不等式的文本(一组或多组)，例如：

```
Equation
   cost      'define objective function' # cost没有域，所以是一个单一方程
   supply(i) 'observe supply limit at plant i' # supply表示在域i上定义的不等式集合
   demand(j) 'satisfy demand at market j';
```

## 2.4.2 GAMS求和或求积符号

### 求和

GAMS的求和语句在求和思想的基础上带有两个参数的操作符，即sum(合计的索引，被加数)，两个参数之间用逗号间隔，其中，第二个参数可以是任何数学表达式(包括求和)，例如：

```
sum(j,x(i,j)) # 1
sum((i,j),c(i,j)*x(i,j)) # 2
sum(i,sum((i,j),c(i,j)*x(i,j))) # 嵌套求和
```

- `# 1` 相当于$\sum_j x_{ij}$

- `# 2` 相当于$\sum_i \sum_j c_{ij}x_{ij}$
- 可以通过使用$符号对求和操作符进行限制，只有满足特定条件的i和j的元素才能包含在求和当中

### 求积

GAMS使用和求和相同的格式定义乘积，只需用prod替代sum即可，例如：

```
prod(j,x(i,j)) # 相当于\prod_{j}x_{ij}
```

- 求和于求积操作符可以在直接赋值语句中对参数进行赋值

```
totsupply=sum(i,b(i)); # total supply over all plants
```

## 2.4.3 定义方程

### 方程定义的组成及其顺序

在GAMS中，方程定义是最复杂的语句，以下是方程定义的组成及其顺序：

- 被定义方程的名字
- 范围(域)
- 约束条件的范围(可选项)
- 符号 ”..“
- 左边的表达式
- 关系操作符
  - 等于：=e=
  - 大于或等于：=g=
  - 小于或等于：=l=
- 右边的表达式

```
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);
```

### Notes

- 用单一gams语句建立过个方程的能力是由域控制的
- 许多现实问题中，一个方程域的其它某些元素需要忽略或区别于其它模式，这时gams可以利用 `$` 或 `such-that` 操作符调节这种结构
- '=e='和'='的区别： `=` 仅在直接赋值时使用，而 =e= 仅用在方程定义
  - 直接赋值在调用求解器之前，赋给参数一个期望数值

        - 直接赋值的所有数学表达式都可以放在右边
    - 方程定义虽然也是描述一个期望关系，但满足于调用求解器之后
    - 重要原则：方程定义必须包含变量，而直接赋值则不必
- 变量可以出现在方程左边或右边或同时在两边，相同的变量在方程中可以出现多次
    - 调用求解器之前，GAMS处理器会自动将方程转化为等价的标准形式，即变量在左边
- 只要方程和它所涉及的所有变量、参数以前被声明过，则方程定义可以出现在GAMS输入的任何位置
    - 参数定义之后，在方程里赋值或重新赋值是被允许的，当使用GAMS输入运行多重模型时是有用的

# 2.5 目标函数

GAMS没有被称为目标函数的外在实体，为了指定最优化的函数，必须创建以个**变量(variable)**，这个变量取任意值(不受符号约束)，是标量(没有范围)，并且出现在等同目标函数的**方程定义**中

# 2.6 模型和求解语句

`模型(model)` 一词在gams表示 `方程的集合` 的含义的，如同其它的gams实体(声明)一样，模型在声明中必须给予一个名字，声明格式如下：关键字**model声明**后紧跟**模型的名称**，再跟着用**斜线括住的方程名称列表**

```
Model transport / all /; # /all/表示包括以前定义的所有方程

Model transport /cost, supply, demand /; # 只写以前定义的需要的方程的名称
```

- 域在列表中忽略了，因为它们不是方程名称的一部分，只有当现有方程的子集组成一个特定生成的模型(子模型)时，才使用域选项

一旦模型已经声明且已经给方程赋值，接下来就可以使用一个求解语句来调用求解器：

- 下面是求解语句的一般格式：
    - 关键字 `solve` ，用于求解声明
    - 求解模型的名称
    - 关键字 `using`
    - 求解程序，以下是常用的求解程序

| 求解程序 | 中文含义 | 求解程序 | 中文含义 |
| --- | --- | --- | --- |
| lp | 线性规划 | nlp | 非线性规划 |

| 求解程序 | 中文含义 | 求解程序 | 中文含义 |
| --- | --- | --- | --- |
| qcp | 二次约束规划 | dnlp | 不连续导数非线性规划 |
| mip | 混合整数规划 | rmip | 松弛混合整数规划 |
| miqcp | 混合整数二次约束规划 | minlp | 混合整数非线性规划 |
| rmiqcp | 松弛混合整数二次约束规划 | rminlp | 松弛混合整数非线性规划 |
| mcp | 混合互补问题 | mpec | 均衡约束数学规划 |
| cns | 约束行非线性系统 | | |

- 关键字 `minimizing` 或 `maximizing`
- 优化变量的名称

```
solve transport using lp minimizing z;
```

# 2.6 显示语句

对于求解器的输出，如最优值，通过 `display` 可以在gams中显示这些结果，如：

```
display x.l, x.m; # l 表示求解变量的最终值，m表示求解变量的边际值
```

# 2.7 '.lo / .l / .up / .m '数据库

gams设计了一个小型数据系统，用来维护变量和方程的记录， `.lo / .l / .up / .m` 是每个记录最重要的 字段 ， 其中 `.lo`表示下限 ， `.up`表示上限 ， `.l`表示水平值或初值或最终值 ， `.m`表示边际值或对偶值 ， 格式：变量跟着字段名，如果需要可在字段中加入域或者域的某个元素

## 2.7.1 变量的边界和初始值的赋值

变量的上下界是根据变量类型设置的，但是这些边界可以被新gams语句覆盖

```
# 假设capacity(i,j)是一个以前被声明和赋值的参数
# 以下语句必须出现在variable(变量)声明之后和solve语句之前
# 直接赋值的所有数学表达式都可以放在右边

x.up(i,j)=capacity(i,j);
x.lo(i,j)=10.0;
x.up('seattle','new-york')=1.2*capacity('seattle','new-york');
```

- Note:
  - gams用户完全控制 `.lo / .up` 字段，而 `.l / .m` 字段是用户初始化，之后由求解器控制
  - 非线性规划中，通过指定尽可能小的上下限范围来帮助求解器是重要的，同时，通过指定初始求解方案对求解器寻找最优值也是很有用的
  - 默认的初始值为0，除非0不再边界范围内

## 2.7.2 最优值的转换和显示

Sovle语句调用最优化程序后，计算出的初始值和边际值的数值放在了数据库的 `.l / .m` 字段中，这可以通过gams的 `display` 语句显示出来

```
# 计算各个市场的需求占每个工厂的百分比

parameter pctx(i,j) percent of market j's demand filled by plants i;
pctx(i,j)=100.0*x.l(i,j)/b(j);
display pctx;
```

Note: 涉及边际值的内容，参考<用户指南> ———— 2.10.2 最优值的转换和显示

# 2.8 GAMS输出

- 输出的部分内容包括：返回输出、错误信息、引用映射、模型统计、方程列表、状态报告、求解报告
- GAMS试图迅速寻找出错误并使它们鹅影响最小化

## 2.8.1 返回输出

GAMS运行输出的第一部分：输入文件的返回或拷贝

- 无论是否出现error，这一步都会执行
- 左边是返回内容的行号
- `$title` 语句使随后的文本内容在顶部输出
- `$offupper` 语句包括混合大小写字母
- `$ontext` 和 `$offtext` 语句之间的内容作为文本用于注释或说明

## 2.8.2 错误信息

当gams编译器在输入文件里遇见错误时，会在 `lst文件` (lst是gams输出文件的后缀名， `gms` 是gams的程序语句文件的后缀名)中的返回输出的出错行处插入编码错误的信息，这些错误信息具有固定的显示格式：在出错语句的下一行，以 **** 开始，之后紧跟 `$` 符号， `$` 紧跟一个 `数字错误代码` ，之后另起一行对这个 `错误代码详细说明`

- 常见错误类型：
  - a. 集合元素的命名不能和gams的保留字符冲突，如不可以是sum
  - b. 省略直接赋值或方程定义前的分号
  - c. 拼写错误
  - d. 数学上的不一致，如：不是$\sum_i x_{ij}$ 对所有的i，应该是$\sum_j x_{ij}$ 对所有的i

```
# a
set s season time /spring,sum,fall,winter /;

# b
# 对于缺少分号，可以通过浏览对照表中的c条目和赋值状况来发现
Parameter c(i,j) 'transport cost in thousands of dollars per case' # 省略了分号
c(i,j) = f*d(i,j)/1000;

# c
Set
   i 'canning plants' / seattle,  san-diego /

Parameter
   a(i) 'capacity of plant i in cases'
        / seatle    350   # 是seattle，不是seatle
          san-diego  600 /

# d
supply(i).. sum(i, x(i,j)) =l= a(i);
```

## 2.8.3 引用映射

- 引用映射(Reference Map)是检测到错误后的最后一部分输出，它是一对引用映射，它包括对输入文本的摘要和分析，对是调试模型和建立文件
- 第一个引用映射是一个对照映射(cross-reference map)，它是模型的所有实体(集合、参数、变量、方程)按字母顺序的对照表，这个对照表显示了每个实体的类型和在输入中实体每个状态的编码引用状况，只需在程序语句中添加下面的语句命令即可

  ```
  $onSymXRef
  ```

  - 引用映射的好处之一是发现用由于标点符号或语法错误而错误输入模型的多余实体
- 引用映射的第二部分是一个按类型分组和相关文本排列的模型实体列表( gdx文件 中可以查找)

## 2.8.4 方程列表

### 方程列表( Equation Listing )

方程列表可用于研究GAMS是否生成了想要的模型
gams中，输入约束通常是一般约束，而在输出约束中是特定约束(对于每个一般方程的默认输出是三个特定方程的约束最大值，如是325而不是比325小的值)，不过，可以通过下列语句改变默认输出个数：

```
option limrow=r; # r是希望的行数，可是以任何数字
```

### 纵向列表( `Column Listing` )

纵向列表类似于方程列表，它对每个一般变量它会显示相应的特定变量的系数
可以通过下面语句改变每个一般变量的特定纵向输出的默认数值：

```
option limcol=c; # c是希望的列数，可是以任何数字
option limcol=r, limcol=c; #可以合并
```

## 2.8.5 模型统计

- 调用求解器之前，gams会在 `Model Satistic` 部分输出一组关于模型规模统计：
  - block计数指一般方程和变量的数目
  - single计数指在特定模型实例中阐述的单独行列数
- 对于非线性模型，还有其它的统计来描述问题的非线性程度

## 2.8.5 状态报告

求解器执行后，gams在 `Solution Report` 部分会输出一份简要的求解摘要，求解器状态(SOLVER STATUS)和模型状态(MODEL STATUS)是最重要的条目

- 期望的求解状态是 1 Normal Completion
- 可能的模型状态有11个，包括:
  - 一般线性规划的终止状态
    - 1 OPTIMAL
    - 3 UNBOUNDED
    - 4 INFEASIBLE
  - 非线性规划的状态
    - LOCALLY OPTIMAL
  - 整数规划的状态
    - 8 INTEGER SOLUTION(即找到一个可行的整数解)

## 2.8.6 求解报告

检查完求解器状态和模型状态后可以检查最优化结果( `Optimal Solution: solEQU和solVAR` )，最优化结果会根据特定方程的名字进行分组打印输出下限值、水平值、上限值和边际值

- `.` 表示0
- `EPS` 代表epsilon，表示极小但非零，这种情况中，EPS表示退化
- 错误条目的标记： `INFES` 表示不可行解和 `NOPT` 表示错误标记
- 如果问题终止于无界，则行数和烈属标记为 `UNBND`

在求解器求解报告的末尾是非常重要的报告摘要( `Report Summary` )，它给出了非优化、不可行、无界的行列数

### 2.8.7 结果显示

由求解器获得的所有水平值和边际值都保存到gams数据库的 `.l` 和 `.m` 字段中，这些值可以转换并且在任何需要的报告中显示出来，只需要列出要显示的数量(如，x)，gams就会自动格式化并标准一个合适的数组，如：

```
display x.l, x.m;
```

## 小结

# 3. GAMS程序

GAMS是一种程序设计语言，GAMS程序通常使用自带的gams文本编辑器编写输入文件和查阅输出文件

## 3.1 GAMS程序的结构

GAMS程序由一个或多个语句构成，语句的作用在于定义数据结构、初值、数据修改、符号关系(方程)

- 程序语句没有固定的顺序，但数据修改有一定的顺序
- 符号实体在使用之前需要声明，且需在赋值语句被引用之前赋值
- 每个语句后要使用分号，除了最后一个语句可用可不用

### 3.1.1 GAMS输入格式

- GAMS输入是自由格式，语句可放在一行中的任何位置，多重语句可放在一行上，一个语句可 连续 出现在多行上，如：

```
      statement;

 statement;statement;statement;

 statement;
 statement;
```

- 单个符号或字词之间可以使用空格和行结束形式
- GAMS不不区分大小写字母
- 一行可以有255个字符，可以插入空白行
- 星号 `*` 和 `$` 用在行首表示非GAMS程序语言输入
  - 以 `*` 开始的行是注释行
  - 以 `$` 开始的行表示此行剩下的部分为编译器选项
    - `$include` 可以实现文件的输入，例如： `$include file_name` 表示在这个语句的位置插入指定文件的内容

## 3.1.2 GAMS语句的分类

GAMS中的每个语句可以被分为两组：声明和定义(赋值)语句和执行语句

### 声明和定义语句

声明语句描述语句(符号)的种类，然后对其赋值或定义

- 下面是常见的声明和定义语句：
  - set、alias、acronym
  - parameter、scalar、table
  - variable
  - equation
  - model

### 执行语句

执行语句是完成行动的指令(集)

- 下面是常见的执行语句：
  - solve
  - display
  - loop、while、for
  - option、assignment、repeat、abort、execute、

# 3.1.3 GAMS程序的组织

GAMS程序语句顺序虽然自由，但是有两种常用的顺序

- 第一种(推荐)
  - 数据
    - **集合声明和定义**
    - **参数说明和定义**
    - 赋值
    - 显示输出
  - 模型
    - **变量声明**
    - **方程声明和定义**
    - **模型声明和定义**
  - 模型求解
    - **求解**
    - **显示输出**
- 第二种
  - 模型
    - 集合声明
    - 参数声明
    - 变量声明
    - 方程声明和定义
    - 模型声明和定义
  - 数据
    - 集合定义
    - 参数定义
    - 显示输出
  - 模型求解
    - 求解
    - 显示输出

```
#第一种
set c 'crop' /wheat, clover, beans/; # 声明并定义
parameter yield 'crop yield' /wheat 1.5
                             clover 6.5
                             beans 1.0/;

# 第二种
# 声明
set c 'crop'; # 声明了标识符c是一个集合
parameter yield 'crop yield';

# 定义
set c /wheat, clover, beans/;# 定义集合中的各个元素
parameter yield /wheat 1.5
                clover 6.5
                beans 1.0/;
```

# 3.2 数据类型和定义

gams有五种基本的数据类型，分别是 `set`、`parameter`、`variable`、`equation`、`model` ， 每个标识符(符号)必须要声明为其中的一种，值得注意的是， `scalar`和`tables` 不是单独的数据类型，而是声明某个符号是parameter的一种快速方法，并且用一个**特有的格式初始化**数字格式的**数据**
五种基本的数据类型的格式：

```
> parameter         a      (i,j)     comment   # Note:域的列表和文本是可选的
> 数据类型关键字     标识符    域列表       文本

> variable x,y,z; # 许多标识符用一个语句声明，标识符之间用逗号隔开
```

# 3.3 语言条目

GAMS的基本符号又被为语句要素，是gams语言的基本结构要素，是辨认和书写gams语句的基本规则，它们分别是 字符、分隔符、变迁、保留字和标记、注释、文本、数字、标识符和空格

- 字符(characters)
  - gams中存在合法字符和非法字符(不能打印的字符和控制字符)之分，非法字符在gams中只能用在 `$ontext`和`$offtext` 语句之内
  - 对于字符列表可点击本链接查阅
- 保留字(reserved words)
  - gams中的保留字(文字和非文字)是预先设定好的关键字(如，set、parameter、variable、equation、model等)，即不允许用户自己定义，不能作为标识符和标签使用
  - 对于保留字列表可点击本链接查阅

- 标识符(identifiers)
  - 标识符是集合、参数、变量、方程、变量等给定的名字，gams规定，标识符以字母开始，后可接多个字母和数字
  - 要注意的是，用于一个数据类型(集合、参数、变量、方程、变量等)的名字，不可以再用于其它类型
- 标签(labels)
  - **标签是集合元素**，可以加引号也可不加
    - 不加引号的标签中的字符用法受到位置的局限，即不加引号的标签必须以字母或数字开头，且只能跟着字母、数字或者'+''-'字符
    - 加引号的标签中，引号用来划定标签的界限，可包含任何合法的字符，且单双引号均可使用
    - 由于添加引号带来繁琐，多使用不加引号的标签，但是，如果使用关键(如，set等)字作为标签则必须加引号

```
Set
   i 'canning plants' / "seattle",  "san-diego" /
   j 'markets'        / new-york, chicago, topeka/;
```

- 文本(text)
  - 标识符和标签可以与一行描述性的文本联合在一起，并被gams保存(可查阅gdx中的text内容)
    - 文本只能写在一行上
  - 文本可以加引号(单双引号都可以)也可以不加引号
    - 加引号的文本可以包含除引号之外的任何字符
    - 不加引号的文本不能以保留字 `..` 和 `=` 开始，且不能包括分号、逗号、斜杠

```
Set
   i 'canning plants' / seattle,  san-diego /
   j  markets         / new-york, chicago, topeka/;
```

- 数字(numbers)
  - 空格不可以加在数字中(gams是空格为分隔符)
  - gams中，实型和整型数据类型没有区别
  - gams中可以使用拓展范围的数学表达式，
    如 `INF(无穷大)`、`-INF(无穷小)`、`UND(未定义)`、`EPS(任意小)`、 `NA(不可用)` 等特殊符号作为拓展范围的数学表达式
    - 除UND(是某中错误操作之后阐述的错误结果，如被0除)之外都可以像普通数字一样被输入
- 分隔符(delimiters)
  - 语句用分隔符分号( `;` )分开，但是，如果下一个语句用关键字(保留字)开始，则可以不用加分号
  - 逗号(,)和斜线(/)用作数据列表的分隔符，其中，逗号结束一个数据元素，斜线结束一个数据列表
- 注释(comment)

- 注释是解释性文本，gams不会对其进行处理或对其进行保留
- 常见的gams程序注释方法：
  - 单行注释(Single Line Comments)：
    - **第一个字符**以星号( * )开始，这一行剩余的字符会被gams编译器忽略，但会在输出文件中原样输出
    - 使用 `$comment` 定制注释符号，如， `$comment !` ，即叹号开始的所在行是注释

```
* 这里是注释 1

$comment !
! 这里是注释 2
```

  - 块注释(Block Comment)
    - **第一个字符**使用特殊的块分隔符 `$` ，如 `$ontext`和`$offtext` 使gams忽略掉程序中的一个完整部分

```
$onText
This problem finds a least cost shipping schedule that meets requirement
$offtext
```

  - 行外注释(End-of-Line Comments)
    - `$onEolCom` 默认注释符号是 `!!`
    - `$eolCom` 可以自定义注释符号，如 `$eolCom &&` 表示 `&&` 后的内容是注释内容

```
$onEolCom
Scalar f 'freight in dollars per case per thousand miles' / 90 /;  !! 这
Table d(i,j) 'distance in thousands of miles'
            new-york  chicago  topeka
seattle         2.5      1.7     1.8
san-diego       2.5      1.8     1.4; !! 这里是注释2

$eolCom &&
Parameter c(i,j) 'transport cost in thousands of dollars per case';
c(i,j) = f*d(i,j)/1000;   && 这里是注释3
```

  - 行内注释(In-Line Comments)
    - `$onInLine` 默认注释符号是字符对 `/* comment */`
    - `$inLineCom` 可以自定义注释符号，如， `$inLineCom /& &/`

```
$onInLine
cost..      z /*这里是注释 1*/ =e= sum((i,j), c(i,j)*x(i,j));

$inLineCom /&  &/
supply(i).. sum(j, x(i,j)) /&这里是注释 2&/ =l=a(i);
demand(j).. sum(i, x(i,j)) =g= b(j);
```

- 隐藏注释(单行注释)
  - 使用 `$hidden` 可隐藏注释，使注释不在输出文件中显示

```
$hidden 这是一个隐藏注释
Set
   i 'canning plants' / seattle,  san-diego /
   j 'markets'        / new-york, chicago, topeka /;
```

- Outside Margin Comments
  - `$onMargin`和`$offMargin` 语句白噢是边框内的语句被执行，边框外的语句被忽略
  - `minCol` 用于定义被识别语句的开始列位置， `maxCol` 用于定义被识别语句的结束列位置

```
$ontext
12345678901234567890123456789012345678901234567890123456789012345678901234567890
$offtext

$onMargin minCol 20 maxCol 45
Now I have          Set i plant /US, UK/     This defines i
turned on the       Scalar x / 3.145 /       A scalar example.
margin marking.     Parameter a, b;          Define some
                                             parameters.
$offMargin
```

# 4. 集合定义

# 5. 数据输入

# 15. Put输出工具

put输出工具的用途在于在格式控制下输出个别项目到不同类别的文档中，如，excel文档

# 15.1 Put语法的基本结构

```
# 定义一个多多个要写入的文件
# fname是gams模型中外部文件的名称
file fname;

# 指定其中一个文件作为要写入的当前文件
put fname;

# 当前文件的实际输出项
# item是输出的任意类型，如文本、标签、参数、变量、方程等
put item;
```

例子：放在模型的末尾输出报告

```
# 定义名为fac.csv的外部文件在gams内部中的文件名称为factors
#这里指定了两个文件
file factors /fac.csv/, results /res.csv/;


# 指定fac.csv为当前可写入的文件
# 注意这里的语句只能是内部名称
put factors;

# 使用put语句将文本内容Transportion Model Factors写入文档
# 不可以省略文本引号(单引号)
# 短斜线代表回车(一条两条三条都可以)
put 'Transportion Model Factors' ///

# f表示标量
#输出项目用分隔符逗号或空格隔开，但是逗号是更强的格式，可以排除任何歧义
'freight cost', f,

# 通过语句中的光标控制符号#和@可以把光标定位到紧跟其后的数字所表示的列和行
@1#6, 'plant capacity'/;  # 把光标定位到第六行第一列，另写一个文本项

# put语句用分号结束
loop(i,put@3,i.tl,@15,a(i)/);

put /'market demand'/;
loop(j,put@3,j.tl,@15,b(j)/);


put results;
put 'Transportion Model Results' //;
loop((i,j), put i.tl,@12,j.tl,@24,x.l(i,j):8:4/);
```

1. 官网：https://www.gams.com ↩
2. 《可计算一般均衡模型的基本原理与编程》GAMS程序：http://www.hibooks.cn/gao/kejian.asp ↩
3. 本节使用的语句来自于 `A Transportation Problem` ，由Dantzig贡献，可在GAMS的Model Library中下载 ↩