CHAPTER 3. INTRODUCTION TO GAMS

I have used the following references. Download them

http://spot.colorado.edu/~markusen/teaching_files/applied_general_equilibrium/GAMS/intro1.pdf

http://spot.colorado.edu/~markusen/teaching_files/applied_general_equilibrium/GAMS/intro2.pdf

1. Starting with GAMS

You are working on a project. Create a project directory on your hard disk
D:\PPD\GAMS\Test1
Thus, you will not spoil the directory where GAMS is located, with your works.

You will use the GAMS-IDE editor to write our programs. You use the editor to enter your GAMS code and to save a file with a name of your choice and the extension **GMS**. The **GMS** file, also called the input file, is then submitted to the GAMS system, which compiles and executes the GAMS code. GAMS creates an output file with the name you chose for the input file followed by the extension **LST**. This latter file is sometimes called the listing file.

A key idea of GAMS-IDE is the project file. In practice, you can think of the project file as a place holder for files and options just like a directory on your hard disk drive. Indeed, the project file is, in some sense, redundant as it just points to your project directory where you have your project files.

Choose **File | Project | New Project**

Find the location of your project folder.

In the box **File name** enter a project name and then click **Open**. GAMS-IDE then creates a project file with the extension **gpr** and stores this file in your project folder.

You are now ready to both create your own GAMS file, to open an existing file, and to open the GAMS model library. We will use a model from the library to keep the introduction simple.
Choose **File | Model Library | Open GAMS Model Library**. This opens a window with a list of GAMS models. Choose **THREEMGE** and GAMS-IDE opens a tab with the GAMS code and adds the file to the project file.

First we will save the file under a new name to keep a copy of original error-free code.
Choose **File | Save as**
and then a file name, for example **THREEMGE1**.
Then delete the letter "E" in the GAMS keyword **SET** (line 15 in the code) and hit the **run** button (the icon with the red arrow).

Move to the top of the process window and double-click the very first red line. This takes you to the place in the input file (the one with the extension **GMS**) where the error was made.

GAMS offers more help if you look at the listing file (the output from the GAMS run stored in the file with the extension **LST**). Click on the tab **THREEMGE1.lst** and make sure that you are at the top of the file. The first part of the listing file is an echo print of the input file. If there are errors in the input file, GAMS adds a coded error messages in the line immediately after the lines with errors. The coded error messages consist of ******** in the first column of the line and is followed by **$** (a dollar sign) below the place in the line above where GAMS thinks there is an error. After the dollar sign follows one or more numerical error codes. These error codes are explained after the echo print and indicates what GAMS thinks is the problem.

You can avoid having to jump to the end of the listing file to find the explanation of the error code. This is done by setting the option **errmsg**. Choose **File | Options** and then the tax **Execute**. Tick **Use following additional parameters** and add **errmsg=1** in the parameter window. GAMS will now add the explanation of the error code(s) immediately after they occur.

Now that we have introduced options in GAMS, we will also mention a few of the other settings that you can control. The first tab **Editor** contains a number of options for the editor in GAMSIDE. If you like to use tabs when programming to help format your input code

nicely and you want to avoid that the tabs are replaced with spaces when GAMS is run, you should make sure that **Insert Tab** is selected in the window for **Tab key action**.

The fourth tab in the options window is **Solvers**. The rows are the list of solvers available in the GAMS system and the columns are the problem types that GAMS can solve. The table tells you what solvers are available for what problem types (indicated by a small square) and what the current selection or default is (indicated by an X). Also, the first column tells you if you have a full license to the solvers or if you have a demonstration license.

GAMS is very relaxed about style and punctuation. For example, GAMS ignores blanks and case. Tabs are also ignored, except when data is entered using tables where tab stops are assumed to have a length of 8 characters.

Comments are added in lines beginning with a ***** in the first column and several lines may be inserted as long as the first line begins with a *****.

2. Example: A single market equilibrium model

There are two equations, supply and demand, and two variables, price and quantity.

GAMS' approach is to formulate the equations and variables as a complementarity problem. This involves (a) associating each equation with a particular variable, called the complementary variable. (b) if the variables are restricted to be non-negative (prices and quantities), then the equations are written as weak inequalities. If the equation holds as an equality in equilibrium, then the complementary variable is generally strictly positive. If the equation holds as a strict inequality in equilibrium, the complementary variable is zero.

Consider first supply of good X with price P. The "supply curve" in simple competitive models is just the firm's (or industry's) marginal cost curve. The supply curve exploits the firm's optimization decision, which involves equating price with marginal cost: P = MC.
However, the good might not be produced since marginal cost might exceed price in equilibrium.
The correct condition for equilibrium is:

$MC \geq P$ with the complementarity condition that $X \geq 0$

If the equation holds with equality, X is (generally) strictly positive; if it holds with strict inequality, then X is 0. Note that the *price* equation is complementary with a *quantity* variable.

Let the cost function for X be given by $\text{COST} = aX + (b/2)X^2$. Marginal cost is given by MC = a + bX. The equation for supply, which is associated with the quantity variable X is then:

$a + bX \geq P$ complementary with $X \geq 0$

.

Now consider demand. Optimizing consumer utility for a given income and prices will yield a demand function of the form X = D(P, I) where I is income. But we will also need to treat the demand side of the model in a complementary fashion, since some goods might be free (P = 0) in equilibrium. The complementarity formulation for the demand side of the model is thus:

$X \geq D(P, I)$ with the complementary condition that $P \geq 0$.

If the weak inequality holds as an equation, P with be strictly positive; if it holds as a strict inequality in equilibrium, X is free and P = 0. Note that the *quantity* equation is complementary with a *price* variable.

In textbook partial-equilibrium models with income suppressed, we just use the slope and intercept parameters for this function: X = c + dP where c > 0, d < 0. Our inequality is then:

$X \geq c + dP$ complementary with $P \geq 0$.

We now have our model: two equations in two unknowns, each associated with the correct complementary variable.

3. Coding the example in GAMS

Comment statements can be used at the beginning of the code, preceded with a *.

```
* M0A.GMS introductory model using MCP
* simple supply and demand model (partial equilibrium)
```

First, comes the list of parameters. If there is text after the parameter with no comma, GAMS does not interpret this text. GAMS saves the text and uses it when parameters are displayed. A semi-colon must be used at the end of the list of parameters.

```
PARAMETERS
```

```
A intercept of supply on the P axis (MC at Q = 0)
B change in MC in response to Q, this is dP over dQ
C intercept of demand on the Q axis (demand at P = 0)
D response of demand to changes in price, dQ over dP
TAX a tax rate used later for experiments;
```

Parameters must be assigned values before the model is solved. Each assignment statement is separate and so eachmust end in a semi-colon. These parameter values are referred to as Case 1 below.

```
A = 2;
C = 6;
B = 1;
D = -1;
TAX = 0;
```

Now we declare a list of variables. They must be restricted to be positive to make any economic sense, so declaring them as "positive variables" tells GAMS to set lower bounds of zero on these variables. The list must end with a semi-colon. Again, it is optional to add descriptive text after the set labels.

```
POSITIVE VARIABLES
P
X;
```

Now we similarly declare a list of equations. We can name them anything we want provided it is a name not otherwise in use or, of course, a keyword. The list ends in a semi-colon.

```
EQUATIONS
DEMAND
SUPPLY;
```

Now we specify our equations. The format is to give the equation name followed by two periods (full stops). Then after a space, the equation is written out in the following way, with =G= GAMS code for "greater than or equal to".

```
SUPPLY.. A + B*X =G= P;
DEMAND.. X =G= C + D*P;
```

Next we need to declare a model: a set of equations and unknowns (above, we could have declared more equations and variables than we are actually going to use in the model, so we need to tell GAMS what exactly is the *n*x*n* system that we want to solve). The format is the keyword *model,* followed by a model name of your choosing. Here we use *equil* for our name. Next comes a "/" followed by a list of the equation names: each equation ends with a period followed by the name of the complementary variable. Each *equation.variable* is followed by a comma before the next one. The model declaration ends with another "/" and a semi-colon.

```
MODEL EQUIL /SUPPLY.X, DEMAND.P/;
```

Finally, we need to tell GAMS to solve the model and what software is needed (GAMS does many other types of problems, such as optimization). Here is the correct statement (MCP stands for mixed complementarity program).

```
SOLVE EQUIL USING MCP;
```

This example uses parameter values which generate an "interior solution", meaning that both X and P are strictly positive. But other outcomes are also possible, and these are both important and common in general equilibrium modeling.

Case 2: It is often the case that a good, or rather a particular way to produce or obtain a good is too expensive relative to some alternative so that this production or activity is not used in equilibrium. We sometimes say that this activity is "slack" in equilibrium. This case, where X=0, can be generated by the following parameter value of the supply (marginal cost) function. There is no need to respecify the model: just change the value of A and repeat the solve statement.

```
A = 7;
SOLVE EQUIL USING MCP;
```

Case 3: The final possibility is that a good or factor of production may be so plentiful that it commands a zero price in equilibrium. This case, where P=0, can be generated by the following parameter value of the supply (marginal cost) function.

```
A = -7;
SOLVE EQUIL USING MCP;
```

4. Comparative static experiments

In the following example, we use a parameter called "TAX", an ad valorem tax on the input (marginal cost) to production. To do so, we declare and specify an additional equation "SUPPLY2" and model "EQUIL2" (GAMS does not allow us to simply rewrite the equation or use the old model name once we name a new equation). Three parameters are declared and used to extract information after we solve the model.

```
PARAMETERS
CONSPRICE consumer price
PRODPRICE producer price (equal to marginal cost)
TAXREV tax revenue (note tax base is producer price);

EQUATIONS
SUPPLY2;

SUPPLY2.. (A + B*X)*(1+TAX) =G= P;

MODEL EQUIL2 /SUPPLY2.X, DEMAND.P/;
```

Before we solve the model, we need to first return the supply intercept to its original value: parameter changes are kept as permanent unless "undone" in the code. Then we resolve the base case (strictly speaking, this is not necessary, it is just recomputing our first case.).

```
A = 2;
TAX = 0;
SOLVE EQUIL2 USING MCP;
```

Now set the tax at 25% and solve the model.

```
TAX = 0.25;
SOLVE EQUIL2 USING MCP;
```

After we solve the model, it is common to extract output using GAMS code. Note that with the tax specified on inputs, the price being solved for is the *consumer price* (price paid by the consumer) not the *producer price* (price received by the producer). Producer price is the same as marginal cost.

GAMS stores three values for variables: its current value, its upper bound (infinity in our case) and its lower bound (zero in our case). When the modeler wants the value of a variable, you

have to use the notation *NAME.L* where the *L* stands for level. Here is the correct notation for extracting the consumer price, producer price, and total tax revenue. GAMS does not automatically print out the values of parameters in the solution, so you have to ask GAMS to *DISPLAY* these values.

```
CONSPRICE = P.L;
PRODPRICE = P.L/(1+TAX);
TAXREV = PRODPRICE*TAX*X.L;

DISPLAY CONSPRICE, PRODPRICE, TAXREV;
```

5. Reading the output

There is a lot of "stuff" in GAMS listing (output) files, which are stored as *MOA.LST* after you run the model (*LST* is for "listing file"). Here are just the relevant parts of our model runs. But first, for any/all solve statement, check to make sure that you see the following (line 132).

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

     MODEL   EQUIL
     TYPE    MCP
     SOLVER  PATH                FROM LINE  30

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
```

Case 1: The *LEVEL* is the solution value of the variables. *MARGINAL* indicates the degree to which the equation corresponding to the variable is out of equality. For *P* (price), the equation is *DEMAND* and the value of the marginal is supply minus demand. For *X* (quantity), the equation is *SUPPLY* and the value of the marginal is the excess of marginal cost over price. Variables that have *positive values* in the solution should have *zero marginals*. Variables that have *zero values* in the solution should have *positive marginals*. Line 184.

```
          LOWER    LEVEL    UPPER   MARGINAL

---- VAR P         .     4.000      +INF       .
---- VAR X         .     2.000      +INF       .
```

Case 2: This is the zero-output case. The price equation holds, but the quantity equation is slack. The marginal of 1.0 indicates that, at the solution, marginal cost exceed price by 1.0. Line 305.

```
            LOWER    LEVEL    UPPER   MARGINAL

---- VAR P         .     6.000      +INF       .
---- VAR X         .       .        +INF     1.000
```

Case 3: This is the free-good case. Now the price equation is slack, and the marginal of 1.0 indicates that, at the solution, supply exceeds demand by 1.0. Line 426

```
            LOWER       LEVEL       UPPER       MARGINAL
---- VAR P  .           .           +INF        1.000
---- VAR X  .           7.000       +INF        .
```

Tax example: Here is the tax example (after the supply intercept is set back at 2.0). Note the "tax incidence" is split between the producer and consumer: the initial price was 4.0. Lin 664

```
                LOWER    LEVEL    UPPER   MARGINAL

---- VAR P           .     4.444      +INF         .
---- VAR X           .     1.556      +INF         .
```

The first section of the GAMS output is an echo print, or a copy, of your input file. GAMS adds line numbers to your code for future reference. Also, if there are compilation errors in your GAMS model, they will appear in this section. Lines 6 to 67

Next follows a reference map. This is not important to beginners, so it suffices here to say that it is an organized presentation of all the entities in the model, including cross-references. For example, the map provides an answer to question: What parameters are in the model and where are they declared, assigned values and referenced? Lines 69 to 71

The listing of equations is probably more useful. This section shows how GAMS interprets your model and includes the current values of sets and parameters installed into your algebraic model. Therefore, this is the section to look for an answer to the question: Did GAMS generate the model you intended? Lines 80 to 87

The model statistics tells you how big your model is. For example, here you can read how many equations and how many variables there is in your model. Lines 114 to 126

The status report is where it gets really interesting. Here you will find an answer to the question: Did your model solve? You should be cheering if it says **NORMAL COMPLETION** in **SOLVER STATUS** and **OPTIMAL** in **MODEL STATUS**. If there is a report of **ITERATION INTERUPT** or **INFEASIBLE** then there is more work to do. Lines 132 to 175

The final default output is a solution report. Here you will find a listing of your results. At the end of the listing file you may also find the results of any display of GAMS symbols you may have added after the solve statement. Lines 180-188.

Finally, you can read on line 682

```
----     61 PARAMETER CONSPRICE          =        4.444  consumer price
            PARAMETER PRODPRICE          =        3.556  producer price (equal
                                                         to marginal cost)
            PARAMETER TAXREV             =        1.383  tax revenue (note tax
                                                         base is producer price)
```