A Primer on GE Modeling with GAMS

David Roland-Holst
UC Berkeley
dwrh@are.berkeley.edu

June, 2009

This note provides a brief introduction to a software tool which is often used in CGE modeling. Generalized Algebraic Modeling System (GAMS) is a high-level programming language which allows nonspecialist computer users to specify and implement economywide models calibrated to datasets of the kind described in the preceding chapters.[1] While the GAMS language is relatively easy to learn for a computer-literate individual with some knowledge of linear algebra, it is flexible enough to implement very large models on a PC platform.[2] In the following discussion, the GAMS language will be introduced via a practical example, i.e. the specification and calibration of a simple CGE specification.

One of the first computable general equilibrium (CGE) models was that of Johansen (1960). A Johansen-style CGE model is written as a system of equations linear in proportional changes of the variables. CGE models of this form include Taylor and Black (1974), Dixon et al. (1982), and Deardorff and Stern (1986).

Perhaps the best-known analytical statement of this type of model was given by Jones (1965). In fact, the Jones algebra and the Johansen CGE formulation are completely analogous techniques. Both are designed to solve nonlinear equation systems by using local first-order approximations. GAMS will be applied to an elementary example of the Johansen-Jones approach to general

1 GAMS is the property of the GAMS Development Corporation, 1217 Potomac Street NW, Washington DC 20007, who has sole authority over its use and entitlement to fees derived therefrom. See Brooke, Kendrick, and Meeraus (1988) for more details.

2 See e.g. Devarajan et al (1994), and Lee and Roland-Holst (1994) for models in excess of 10,000 equations.

equilibrium modeling. We first set out the Jones algebra and then describe its translation into the GAMS language. An appendix gives the full listing of the GAMS program.

The Jones Algebra

Consider a two-sector model with the following production structure $Y_j=F_j(L_j,K_j)$, where j=1,2, the first sector produces importable goods (Y_1), and the second produces exportable goods (Y_j).[3] The factors are defined by L_j, labor input into sector-j production, with $L_1 + L_2 = L$, where L is the employment level, and K_j, capital input into sector-j production, and $K_1 + K_2 = K$, where K is the current stock of capital.

In order to formulate a complete model, more notation is needed. Let w denote the wage rate, r the capital rental rate. Now let p_j and p_{wj} denote the domestic and world prices of good j, respectively, while a_{ij} is the input coefficient for input i into the production of good j. Finally, let t_1 denote an import tariff and s_2 is an export subsidy.

This notation and the assumptions of constant returns to scale in production and perfect competition yield the following general equilibrium system:

Fixed-employment conditions:

$$a_{L1}Y_1 + a_{L2}Y_2 = L \qquad (6.1)$$

$$a_{K1}Y_1 + a_{K2}Y_2 = K \qquad (6.2)$$

Average-cost pricing conditions:

3 It is possible to include both nontraded and intermediate goods into the Jones algebra. For an example of this, see Tobey and Reinert (1991).

$$wa_{L1} + ra_{K1} = p_1 \quad (6.3)$$

$$wa_{L2} + ra_{K2} = p_2 \quad (6.4)$$

Conditional input coefficient functions:

$$a_{L1} = a_{L1}(w,r) \quad (6.5)$$

$$a_{L2} = a_{L2}(w,r) \quad (6.6)$$

$$a_{K1} = a_{K1}(w,r) \quad (6.7)$$

$$a_{K2} = a_{K2}(w,r) \quad (6.8)$$

Domestic price equations:

$$p_1 = (1+t_1)p_{w1} \quad (6.9)$$

$$p_2 = (1+s_2)p_{w2} \quad (6.10)$$

In the above ten-equation system, the exogenous variables are L, K, p_{w1}, and p_{w2}, the endogenous variables are Y_1, Y_2, a_{L1}, a_{L2}, a_{K1}, a_{K2}, w, r, p_1. The terms p_2. t_1 and s_2 are parameters.

In order to put the equations into proportional change form, we need to introduce some additional notation. The circumflex, "^", denotes percentage change in the indicated variable. The parameter λ_{ij} denotes the proportion of factor i used in sector j, while θ_{ij} denotes the share of factor i in the output of sector j and σ_j denotes the elasticity of substitution between labor and capital in sector j.

With these conventions in mind, one can obtain the following system of equations by total

differentiation of equations (6.1)-(6.10):[4]

$$\lambda_{L1}\hat{Y}_1 + \lambda_{L2}\hat{Y}_2 = \hat{L} - \lambda_{L1}\hat{a}_{L1} - \lambda_{L2}\hat{a}_{L2} \tag{6.11}$$

$$\lambda_{K1}\hat{Y}_1 + \lambda_{K2}\hat{Y}_2 = \hat{K} - \lambda_{K1}\hat{a}_{K1} - \lambda_{K2}\hat{a}_{K2} \tag{6.12}$$

$$\theta_{L1}\hat{w} + \theta_{K1}\hat{r} = \hat{p}_1 \tag{6.13}$$

$$\theta_{L2}\hat{w} + \theta_{K2}\hat{r} = \hat{p}_2 \tag{6.14}$$

$$\hat{a}_{L1} = \theta_{K1}\sigma_1(\hat{r} - \hat{w}) \tag{6.15}$$

$$\hat{a}_{L2} = \theta_{K2}\sigma_2(\hat{r} - \hat{w}) \tag{6.16}$$

$$\hat{a}_{K1} = \theta_{L1}\sigma_1(\hat{w} - \hat{r}) \tag{6.17}$$

$$\hat{a}_{K2} = \theta_{L2}\sigma_2(\hat{w} - \hat{r}) \tag{6.18}$$

$$\hat{p}_1 = \hat{p}_{w1} + dt_1/(1+t_1) \tag{6.19}$$

$$\hat{p}_2 = \hat{p}_{w2} + ds_2/(1+s_2) \tag{6.20}$$

GAMS Implementation

The next step is to translate the system 6.11-6.20 into the GAMS programming language. Before doing so, some discussion of solution strategy, arithmetic operators, and relational operators is warrented. Since the above system is linear and square (number of endogenous variables equals number of equations), it can be solved by matrix inversion, which is how Johansen models generally have been solved.[5] The GAMS software, however, was designed to solve more general linear and nonlinear programming problems. We adapt it to exactly determined CGE models by simply

4 Equations 6.13 and 6.14 require the application of the envelope theorem. See Jones (1965).

5 See Dervis, de Melo, and Robinson (1982), Appendix B.3.

specifying the model's equations as the system of constraints and including an arbitrary objective function. The latter is superfluous since a fully specified general equilibrium model should have a unique solution.[6]

Like most programming languages, GAMS has a variety of operators. These are divided into three principal groups, arithmetic, relational, and conditional. The arithmetic operators used in GAMS are of the following:

```
**        exponentiation
* /       multiplications and division
+ -       addition and subtraction
```

These are listed in order of the precedence which would be applied in the absence of parentheses. Exponentiation is performed first, and multiplication and division precede addition and subtraction. Finally, computation proceeds from right to left through an open (i.e. parentheses-free) expression.

Relational operators used in GAMS are as follows:

```
lt, le, eq, ne, ge, gt
     less than, less than or equal, not equal, etc.
not            not
and            and
or xor         or, either or
```

6 The optimization features of GAMS have been used by a number of authors to study policy responses to changing eonomic conditions. See e.g. Lee and Roland-Holst (1993).

These again are listed in order of open precedence. Liason between arithmetic and logical operations is provided by the usual zero-false, nonzero-true standard.

GAMS programs consist of a series of statements followed by semicolons:

```
Statement ;
     .
     .
     .
Statement ;
```

GAMS programs are commonly structured as follows:

```
Data:
     SAM
     Parameters and other data

Definitions:
     Sets
     Parameters
     Initial values
     Variables
     Equations

Model:
Solution:
Display:
```

We begin the example general equilibrium program with the data input. This data consists of two components, a social accounting matrix and supporting tables of structural parameters and other data. Generally, these components are loaded into the model from two separate files with the GAMS "include *filename ;*" statement. In the present, simpler Jones model, we omit these two files an proceed directly with definitions. The first type of definition is a set declaration, which generally take the form:

```
sets
     setname name1   text  /elements/
                .
                .
                .
     setname namen text  /elements/
     ;
```

In the example model, indices are required for sectors and factors, which leads to the following sets definition:

```
sets
     i         industries  / 1*2 /
     f         factors / L,K /
;
```

Note that there are two ways to list set elements. They may be listed individually, separated by commas, or they may be listed as a range, indicated by an asterisk. These two options also can be

used together. For example, a more complex set of elements might be / e1*e10, e12, e14 /.

There are essentially four ways of introducing data into a GAMS program:

1) A scalars statement;

2) A parameters statement with assigned values;

3) A parameters statement without assigned values, followed by a table statement;

4) A parameters statement without assigned values, followed by assignment statements.

The present CGE example will utilize all four means of entering data, beginning with a scalars statement.

A scalars statement can be used to declare and assign a value to a parameter with zero dimension (i.e. not indexed by a set) and takes the form:

```
scalars
     scalar    name1  text  /value/
               .
               .
               .
     scalar  namen  text  /value/
;
```

As mentioned above, a dummy objective function is usually used to solve a square CGE model with the GAMS optimization software. It is often convenient to simply assign this function a constant scalar value as follows:

```
scalars
```

```
        dummy  named  / 1.0 /
;
```

Now consider the parameters statement, which declares and (optionally) assigns values to the parameters of the model. The parameters of the example model are λ_{ij}, θ_{ij}, σ_j, t_1, s_2, dt_1, and ds_2. We assign values for σ_j directly in the parameters statement. Values for λ_{ij} and θ_{ij} will be assigned in table statements. Values for the remaining parameters will be given in assignment statements. In order to illustrate particular GAMS features, we also introduce three further parameters with the labels "tarhat," "subhat," and "cphat."

The parameters statement has the general form:

```
parameters
     parameter  name1  text  /values/
                .
                .
                .
     parameter namen  text  /values/
;
```

For the Johansen/Jones model, the statement takes the form:

```
parameters
     lambda(f,i)    factor allocation
     theta(f,i)factor income share
     sigma(i)  elasticity of factor substitution
               / 1  0.8
                 2  0.9 /
     t(i)           initial tariff
     s(i)           initial subsidy
     dt(i)          change in tariff
     tarhat(i) proportional change in tariff
     subhat(i) proportional change in export subsidy
     cphat(i)  proportional price change commercial policy
     ;
```

The assignment statements used to enter parameter values have been designed to feature the GAMS dollarsign control character, which can be used in two ways. A $ on the left-hand-side of an assignment statement is a conditional assignment: "[I]f the logical relationship is true, the assignment is made; if it is not, however, the existing value is retained, zero being used if no previous value has been given".[7] A $ on the right-hand-side of an assignment statement implies an if-then-else sequence and an assignment is always made.[8]

7 Brooke, Kendrick, and Meeraus (1988), p.72.

8 The reader might find it useful to think of the $ operator as a "such that" operator.

The assignment statements for the Johansen/Jones model are

```
t('1') = 0.20 ;
t('2') = 0.30 ;
dt('1') = 0.10 ;
dt('2') = 0.15 ;

tarhat(i) $ ( t(i)  gt  0 )  =  dt(i)/(1 + t(i)) ;
subhat(i) $ ( s(i) gt 0 )  =  ds(i)/(1 + s(i)) ;

cphat(i)  =  tarhat(i) $ t(i) + subhat(i) $ s(i) ;
```

The first four statements refer to specific elements of index i, and these elements must be put in single or double quotations. The fifth and sixth statements make assignments to tariff and subsidy proportional change variables, respectively, if the conditions following the dollar operators are true. If the conditions are not true, no assignment is made; the existing value is retained, zero being the default if no previous value was assigned. In the seventh statement, the dollar operators on the right-hand-side of the equation govern which of the two values, tarhat(i) or subhat(i) are assigned to cphat(i). The expressions $ t(i) and $ s(i) are the conditions that t(i) and s(i), respectively, be nonzero.

Next, we will demonstrate how values for λ_{ij} and θ_{ij} can be entered with a table statement. Table statements can come in many different forms, of which only one example is provided here:

```
table  lambda(f,i)

                              1                2
            L                 0.50             0.50
            K                 0.25             0.75
    ;

table theta(f,i)
                              1                2
            L                 0.60             0.40
            K                 0.40             0.60
    ;
```

The first line of a table statement begins with the word 'table'. This is followed by the variable name, including set domains. Labels are used to generate a grid, and values are entered into this grid. Any blanks in the grid denote zeros. It is not necessary to list all elements of a set as row or column labels. Where an element is left out, the corresponding row or column will be a vector of zeros. Labels cannot be repeated, however. The table statement ends with a semicolon. In contrast to the scalars and parameters statements, only one parameter can by initialized in a table statement. Therefore, separate table statements are required for each parameter to be initialized.

Next, we will display the parameters using a GAMS display statement. The important thing to remember about a display statement is that, in listing the parameters to be displayed, set domains are not included. For the present example, the parameters are displayed using the following statement:

```
display        lambda, theta, sigma, t, dt ;
```

This completes the data component of our GAMS general equilibrium model. This is generally followed by the model component, which begins with a variables declaration statement. The general form of the GAMS variables statement is as follows:

```
variables
     variable  name1  text
               .
               .
               .
     variable  namen   text
     ;
```

In the case of our model, the variables to be declared are the endogenous variables, the exogenous variables, and a dummy variable. These are declared as follows:

```
variables
     yhat(i)   proportional change in production
     ahat(f,i) proportional change in input
     what      proportional change in wage rate
     rhat      proportional change in capital rental rate
     phat(i)   proportional change in domestic price

     lhat      proportional change in labor endowment
     khat      proportional change in capital endowment
     psthat(i) proportional change in world price

     omega          dummy variable for objective function
     ;
```

Equation identifiers are declared in a GAMS program using an equations statement. In general, the equations statement appears as:

```
equations
     equation  name1  text
               .
               .
               .
     equation  namen   text
     ;
```

For the present CGE model, the equations statement is as follows:

```
equations
     fxelab    fixed employment of labor
     fxecap    fixed employment of capital
     acp(i)    average cost pricing
     linp(i)   labor input
     kinp(i)   capital input
     domp(i)   domestic prices
     obj       objective
     ;
```

Next, equations must be defined. This is done in a series of statements. For equations which are equalities, the general form is as follows:

```
equation  name1..  left-hand side =e= right-hand side ;
               .
               .
equation  namen..  left-hand side =e= right-hand side ;
```

Two decimal or period points '..' are required between the equation name and the equation algebra. The '=e=' notation represents the equality sign for equation definitions. It is distinct from the more usual '=' symbol used in parameter assignments. Each equation definition is a GAMS statement and ends in a semicolon. Equation definitions may be indexed in those cases where the variable being determined is defined as a set.

For our model, the definitions are as follows:

```
fxelab..    sum(i, lambda('l',i)*yhat(i)) =e= lhat -  sum(i, lambda('l',i)*ahat('l',i));

fxecap..    sum(i, lambda('k',i)*yhat(i)) =e= khat - sum(i, lambda('k',i)*ahat('k',i));

acp(i)..    theta('l',i)*what + theta('k',i)*rhat =e= phat(i);

linp(i)..   ahat('l',i) =e= theta('k',i)*sigma(i)*(rhat-what);

kinp(i)..   ahat('k',i) =e= theta('l',i)*sigma(i)*(what-rhat);

domp(i)..   phat(i) =e= psthat(i) + cphat(i);

obj..       omega =e= dummy;
```

Note that, when referring to a particular element in an assignment statement or an equation definition statement, the element name is put in quotation marks. The 'sum' function is used to calculate sums over the domain of a set. Its general form is sum(set name, expression). It is used in the first two equation definitions to sum expressions over set i. It is also possible to include a dollar control operator after the set name in a sum function in order to restrict the elements of the set which are included in the summation.

The above set of equations determine the ten endogenous variables and the dummy variable. Still to be specified is the model closure.[9] The closure is given as follows:

9 "(P)rescribing closure boils down to stating which variables are endogenous or exogenous in an equation system" (Taylor, 1990, pp. 15-16).

```
lhat.fx = 0.00 ;
khat.fx = 0.00;
psthat.fx('1') = 0.00;
psthat.fx('2') = 0.00;
```

While a GAMS parameter has a single value associated with it, a GAMS variable has four such values. They are

```
.lo  the lower bound
.up  the upper bound
.l   the activity level
.m   the marginal value
```

The lower and upper bounds are the minimum and maximum values, respectively, that a variable can take on during optimization. The activity level is the current value of a variable, and the marginal value is the effect of the variable value after optimization on the objective function. In cases where the lower and upper bound coincide, the variable is fixed, and the suffix 'fx' is used to assign the fixed value. This is what is done in the above model closure. The first two equations address factor market closure, fixing factor supplies, while the second two equations address external sector closure, fixing world prices. The user can introduce exogenous changes in any or all of these four variables.[10]

Finally, we need a model statement, a solve statement, and a final display statement for the activity levels of the variables after solution. These are as follows:

10 Other types of closures are, of course, possible. For example, Tobey and Reinert (1991) use an export demand

```
model simple /all/;

solve simple maximizing omega using nlp;

display yhat.l, ahat.l, what.l, rhat.l, phat.l, lhat.l,
     khat.l, psthat.l;
```

The model statement declares a model named 'simple' which consists of all the declared equations. The model is solved by maximizing omega. Since omega is set equal to the dummy parameter, the outcome of this maximization procedure is simply to solve the ten constraint equations of the maximization problem for the ten endogenous variables. The term 'nlp' refers to non-linear programing. The solve statement invokes a solver called MINOS. Since the system of equations in our model is linear, it solves very quickly.

Why GAMS?

This module presented a linearized, Johansen-Jones approach to general equilibrium modeling. As we mentioned above, it is possible to solve this class of CGE models using matrix inversion. What, then, is the utility of GAMS? The linearization technique is a local approximation, useful for small changes in exogenous variables. A more general approach to CGE modeling is to specify functional forms, constructing a square but *nonlinear* system of equations. Such a system is not solvable by matrix inversion. For these problems, the GAMS package and the MINOS solver are quite useful.

(..continued)
function to specify rest-of-the world behavior. This replaces the fixed world export price used here.

APPENDIX A: GAMS SIMPLE CGE PROGRAM LISTING

```
$title   a simple general equilibrium model using gams
$offsymlist offsymxref

sets
     i      industries    /1 * 2/
     f      factors       /L, K/
     ;

scalars
     dummy          dummy parameter       /1.00/
     ;

parameters
     lambda(f,i)   factor allocation share
     theta(f,i)    factor income share
     sigma(i)      elasticity of substitution
          /1  0.8
           2  0.9/
     t(i)          initial tariff
     s(i)          initial subsidy
     dt(i)         change in tariff
     ds(i)         change in export subsidy
     tarhat(i)     proportional change in tariff
     subhat(i)     proportional change in export subsidy
     cphat(i)          proportional  change  in  price  due  to
commercial policy
     ;

    variables
     yhat(i)       proportional change in production
     ahat(f,i)     proportional change in input
     what          proportional change in wage rate
     rhat          proportional change in capital rental rate
     phat(i)       proportional change in domestic price

     lhat          proportional change in labor endowment
     khat          proportional change in capital endowment
     psthat(i)     proportional change in world price

     omega         dummy variable
     ;
```

```
equations
     fxelab        fixed employment of labor
     fxecap        fixed employment of capital
     acp(i)        average cost pricing
     linp(i)       labor input equations
     kinp(i)       capital input equations
     domp(i)       domestic prices
     obj           objective
     ;

 * calibration
t('1')  = 0.20;
s('2')  = 0.30;
dt('1') = 0.10;
ds('2') = 0.15;

tarhat(i) $ (t(i) gt 0) = dt(i)/(1+t(i));

subhat(i) $ (s(i) gt 0) = ds(i)/(1+s(i));

cphat(i) = tarhat(i) $ t(i) + subhat(i) $ s(i);


table lambda(f,i)

                    1                   2
            L       0.50                0.50
            K       0.25                0.75
            ;


table theta(f,i)

                    1                   2
            L       0.60                0.40
            K       0.40                0.60
            ;

display lambda, theta, sigma, t, dt;
```

```
* equation definitions

fxelab..    sum(i, lambda('l',i)*yhat(i)) =e= lhat
                        - sum(i, lambda('l',i)*ahat('l',i));

fxecap..    sum(i, lambda('k',i)*yhat(i)) =e= khat
                        - sum(i, lambda('k',i)*ahat('k',i));

acp(i)..         theta('l',i)*what  +  theta('k',i)*rhat  =e=
phat(i);

linp(i)..                                ahat('l',i)         =e=
theta('k',i)*sigma(i)*(rhat-what);

kinp(i)..                                ahat('k',i)         =e=
theta('l',i)*sigma(i)*(what-rhat);

domp(i)..   phat(i) =e= psthat(i) + cphat(i);

obj..       omega =e= dummy;


* model closure (exogenous variables)

lhat.fx = 0.00;
khat.fx = 0.00;
psthat.fx('1') = 0.00;
psthat.fx('2') = 0.00;


* model declaration

options solprint=off;
options iterlim=100,limrow=0,limcol=0,domlim=0;

model simple /all/;


solve simple maximizing omega using nlp;


display yhat.l, ahat.l, what.l, rhat.l, phat.l, lhat.l,
          khat.l, psthat.l;
```