Using IMPLAN Social Accounts for Applied General Equilibrium Modeling

Thomas F. Rutherford*‡ and Andrew Schreiber†‡

‡Department of Agricultural and Applied Economics, University of Wisconsin-Madison, US

May 26, 2016

Contents

1 Introduction

This document serves to elucidate on each element of our build stream for using IMPLAN data in GAMS for running computable general equilibrium (CGE) models. Here, a step by step overview of each element of the build stream is given, along with directions on how to use the included batch file that structures the working directory in accordance to the build stream file associations. For previous implementations, see (Rausch and Rutherford, 2009).

*Email: rutherford@aae.wisc.edu
†Email: schreiber@aae.wisc.edu

The build stream comes with four separate GAMS programs for data reconciliation and a batch file that automatically runs the programs for any given IMPLAN data set(s). These files are:

1. `readimplan.gms`: This program serves to read in the raw data directly from the GAMS files outputted by the IMPLAN software.
2. `condense.gms`: Once the data is read into GAMS, this program runs a series of balancing checks. If the data fail our microconsistency check (row sums do not equal column sums), we apply a least squares balancing routine that induces sparsity.
3. `samgen.gms`: After enforcing standard accounting identities in the data set, this program structures the data into a social accounting matrix conducive to the CGE modeling framework. Additionally, an excel file is outputted for the interested individual.
4. `partition.gms`: Finally, in order to verify the data has been read into the system correctly, this program reads in the excel file, formulates an accounting CGE model, and verifies that benchmark consistency holds (i.e. the model is properly calibrated).
5. `run.bat`: While the above GAMS programs are subject to alterations by the interested modeler, the provided batch file runs everything automatically. However, the individual must be careful to point the program to the correct data directory (directions below).

We progress in the listed order of the programs above by describing the key elements of each program (line by line if need be). The idea behind this technical note largely concerns more of the logical progression of each program rather than understanding every line of GAMS code provided. Notably, when including actual GAMS code from the GAMS programs, we introduce only fractions of the actual code for the sake of brevity. By and large, the "..." symbol is used when more of similar code exists for a given part. GAMS code is written in `blue` text in this document.

2 Reading IMPLAN Data into GAMS

GAMS is fundamentally a set driven mathematical programming language. Before the modeler can do anything in this framework, set indices must be defined and labeled for later use. For instance, in `readimplan.gms`, we've defined IMPLAN sets to be:

```
set r   Rows and columns in the SAM /

*               1*440               Industries,

        1       "Oilseed farming",
        2       "Grain farming",
        3       "Vegetable and melon farming",
        4       "Fruit farming",
        5       "Tree nut farming",
        6       "Greenhouse, nursery, and floriculture production",
        7       "Tobacco farming",
        8       "Cotton farming",
        9       "Sugarcane and sugar beet farming",
        10       "All other crop farming",
        11       "Cattle ranching and farming",
        12       "Dairy cattle and milk production",
        13       "Poultry and egg production",
        14       "Animal production, except cattle and poultry and eggs",
```

```
        15        "Forestry, forest products, and timber tract production",
...
set t   Transfer account codes  /
        3001*3440        Commodities,
        15001               "Corporate Profits with IVA"
        15002               "Emp Comp (Wages/Salary w/o Soc Sec)"
        15003               "Employee Comp (Other Labor Income)"
        15004               "Proprietors Inc (w/o Soc Sec & CCA)"
...
```

The structure of the IMPLAN data can be summarized in Table 1, which categorizes the data into our indexed framework. In this first pass for reading in the data, we are restricted to using the predefined numerical set identifiers for each element. As is later apparent, this is changed for ease in remembering the precise definition of each set element (without having to look up, say, what is identified by set element 10003). Note that each set is labeled `r` or `t` which is a simple way to reference all elements in a row of the social accounting matrix (`r` for row) or transfer accounts (`t` for transfers).

Before continuing on how the data was read into GAMS once sets were defined, we've incorporated a bit of dynamics into the programming that allows the individual to make simple directory changes without having to repeatedly change each applicable spot in the code. For instance, we have:

```
$if not set datadir $set datadir ..\data\Wisconsin
$if not set regions $set regions Adams
$if not set ds $set ds Adams
$set runtrace yes
$if not set samscale $set samscale (1/1000)
```

Notably, for modelers that only wish to deal with the batch program, this bit of information is largely irrelevant. However, if you'd like to run `readimplan.gms` on its own (not in the context of all other programs), then you can change the above information. For instance, if you have data for Cook county, Illinois as opposed to Adams county, Wisconsin, simply change the top three entries to (assuming your directories are set up with the Cook data file in ..\data \Illinois):

```
$if not set datadir $set datadir ..\data\Illinois
$if not set regions $set regions Cook
$if not set ds $set ds Cook
```

However, in most cases, changing elements in individual programs is *not recommended*. If you are somewhat unfamiliar with GAMS programming, identifying errors can be a tedious task indeed.

The next task concerns actually reading the GAMS data file outputted by the IMPLAN software. For a description on how to translate IMPLAN data into GAMS readable data via their software system, go to IMPLAN-Group-LLC. (2014). In order to do so in GAMS, we use the simple command:

```
alias (r,c);

parameter sam(r,t,c)  Base year social accounts /
$offlisting
$include "%datadir%\%regions%.gms"
$onlisting
/;
```

Above, we specify a *parameter* which denotes a data element in GAMS (for information on GAMS elements, see Rosenthal (2004)). The command `alias` allows the modeler to create two identical sets without having to explicitly specify the second set (in this case `c`). The parameter is then specified over

Table 1: Raw Data

ACTIVITY	GAMS INDEX	DESCRIPTION
Sectors:		
	$1 * 440$	Production activities for the 440 sectors in the IMPLAN database
Commodities:		
	$3001 * 3440$	Commodities for productive purposes
Factors of Production:		
	5001	Employee Compensation
	6001	Proprietary Income
	7001	Other Property Income
	8001	Indirect Business Taxes
Private (household) institutions:		
	10001	Households less than 10k
	10002	Households 10-15k
	10003	Households 15-25k
	10004	Households 25-35k
	10005	Households 35-50k
	10006	Households 50-75k
	10007	Households 75-100k
	10008	Households 100-150k
	10009	Households 150k+
Public (government) institutions:		
	11001	Federal Government NonDefense
	11002	Federal Government Defense
	11003	Federal Government Investment
	12001	State Local Govt NonEducation
	12002	State Local Govt Education
	12003	State Local Govt Investment
Corporate institutions:		
	13001	Enterprises (Corporations)
	14001	Gross Private Fixed Investment (Capital)
	14002	Inventory Additions Deletions
Transfers:		
	$15001 * 15056$	Transfer satellite accounts
Trade:		
	25001	Foreign Trade
	28001	Domestic Trade

Note: 1: A list of all 440 IMPLAN industries can be found in (IMPLAN-Group-LLC., 2014).

the relevent set indices `(r,t,c)` and called `sam` for social accounting matrix. The data from the IMPLAN software is then read in using the `$include` command.

In order to verify that the data was read in properly, we provide a series of checks that row and column sums balance. Following the simple balancing checks, we create a mapping to more easily understood set names. For instance:

```
set map(j,*) /

empl.5001   Employee Compensation
```

```
prop.6001   Proprietary Income
othp.7001   Other Property Income
btax.8001   Indirect Business Taxes
hh1.10001   Households LT10k
hh2.10002   Households 10-15k
hh3.10003   Households 15-25k
hh4.10004   Households 25-35k
hh5.10005   Households 35-50k
hh6.10006   Households 50-75k
hh7.10007   Households 75-100k
hh8.10008   Households 100-150k
hh9.10009   Households 150k+
...
```

The example provided above indicates the types of names used for each set element. Rather than having to remember that households with annual income between 25-30 thousand dollars is 10004, we simply relabel this to `hh4` as in the 4^{th} household income grouping. For a complete listing of the names we provide for each set element, refer to Appendix A.

Once the data are labeled in a more memory friendly way, we take the aggregate social accounting matrix and divide it up into its submatrix components. For instance, we have:

```
parameter         make(j,tt,jj)     Domestic industry make matrix
                  use(j,tt,jj)      Domestic industry use matrix
                  iuse(j,tt,jj)     Domestic institutional use matrix
                  fd(j,tt,jj)       Factor input matrix : industry use of factors
                  fexprt(j,tt,jj)   Factor exports
...
```

The above specifies various submatrices that are standard in input output analysis. That is, the *make* matrix specifies the amount of each commodity supplied by a given sector while the *use* matrix specifies the amount of each commodity needed to produce a given sector's output. The set identifiers `(j,tt,jj)` specify the sector and commodity (`j,jj`) and the transfer type, `tt`. IMPLAN data is structured differently than say the national tables in that they provide information in that third dimensional array of transfer types. Later, this information will be stored in satellite tables and the standard two dimensional framework will be employed for CGE analysis. The code then loops through the mappings (set identifiers in Appendix A) to define each submatrix via commands such as:

```
PUTCLOSE 'Relabelling MAKE';
loop(samdomain(s,t,g),
        make(sj,tt,gj)\$jdomain(s,t,g,sj,tt,gj) = sam(s,t,g);
);
unread(samdomain(s,t,g)) = no;
```

The `PUTCLOSE` command writes to the command line (or processing window if using the GAMS IDE) which submatrix is being process. The `loop` command then uses the mapping we've defined between the traditional numerical IMPLAN index and alphabetic index to formulate the submatrices. Once this is finished, a series of checks are performed to ensure that we haven't missed any data elements along the way and that the data adds up to original totals.

The program concludes by defining base year statistics with another mapping mechanism via commands such as:

```
parameter        echop        Echoprint of base year statistics;

echop(acctcat,"make")    = sum((mapcat(acctcat,tt),sj,gj), make(sj,tt,gj));
echop(acctcat,"use")     = sum((mapcat(acctcat,tt),gj,sj), use(gj,tt,sj));
```

```
echop(acctcat,"iuse")    = sum((mapcat(acctcat,tt),gj,ij), iuse(gj,tt,ij));
echop(acctcat,"fd")      = sum((mapcat(acctcat,tt),fj,sj), fd(fj,tt,sj));
echop(acctcat,"imake")   = sum((mapcat(acctcat,tt),ij,gj), imake(ij,tt,gj));
...
```

The above commands sum over the aggregated categories, `mapcat`, which is a mapping that defines categories such as:

```
set        mapcat(acctcat,tt) /
        INCOME.(
                cprf        "Corporate Profits with IVA (15001)"
                ncmp        "Emp Comp (Wages/salary w/o soc sec) (15002)"
                ecmp        "Employee Comp (other labor income) (15003)"
                prop        "Proprietors Inc (w/o soc sec & CCA) (15004)"
                rent        "Rent with capital consumption Adj (15005)"
                btrn        "Business transfers (15006)"
                divd        "Dividends (15007)"
                nint        "Interest (net-from industries) (15008)"
                gint        "Interest (gross) (15009)"
        ),
...
```

The above aggregate tracks total income levels in the various submatrices that have previously been specified.

Next, in order make the data files as clean and compact as possible, all sectors and set elements that are unused for a particular region are dropped (GAMS automatically associates dropped elements with 0 if read back in) and then stored as:

```
PUTCLOSE 'Writing %ds%_data.gdx';
execute_unload
'..\temp\gdx\%ds%_data.gdx',sj=s,gj=g,fj=f,tt=t,tj=trd,ij=i,j,make,
use,iuse,fd,fexprt,imake,fs,trnsfer,fimprt,trnshp,sexport,iexport,simport,iimport;
```

Again, when running this program, the modeler will be able to see via the command prompt when data files are being written via the `PUTCLOSE` command. This data file is then used in the next program, `condense.gms`.

3 Microconsistency

As is similar to the first program, `condense.gms` begins by explicitly specifying all set elements in the IMPLAN data set. Fortunately, because much of the set specification legwork was done in the previous program, we can simply read in set elements from the `*.gdx` data file via:

```
set     f(*)        Factors,
        t(*)        Accounts,
        i(*)        Institutions
        j(*)        Aggregated SAM accounts;

$gdxin '..\temp\gdx\%ds%_data.gdx'
$load f t i j
```

The above bit of GAMS code specifies the sets needed for reading in the benchmark data defined in the previous `readimplan.gms` program. Sectors and goods are read in explicitly. Before continuing, consider Table 2 for a description on the sets used.

TABLE 2: SET INDICES

SET	GAMS INDEX	DESCRIPTION
All Accounts:		
	`t`	Aggregate account index
Sectors and Goods:		
	`s,g`	Sectors and commodities
Factors of Production:		
	`f`	Aggregate factors of production category
Capital Account Factors:		
	`fk`	Factors allocated to the capital account
Institutions:		
	`i`	Aggregate institutions category
Private (household) institutions:		
	`h`	Household accounts
Public (government) institutions:		
	`pub`	Government accounts
Corporate institutions:		
	`corp`	Corporate accounts
Trade:		
	`trd`	Trade accounts

Once we have properly read in all set identifiers, we pull in the actual data parameters that were stored in the `*.gdx`. Again, in order to do this, we simply specify parameter names and read the data in via the `$gdxin` and `$loaddc` commands.

```
PARAMETER
        make_(s,t,g)                Domestic industry make matrix
        use_(g,t,s)                Industry use matrix
        iuse_(g,t,i)                Institutional use matrix
        fd_(f,t,s)                Industry use of factors
        imake_(i,t,g)                Institutional make matrix
...

$loaddc make_=make use_=use iuse_=iuse fd_=fd fexprt imake_=imake
$loaddc fs trnsfer_=trnsfer fimprt trnshp_=trnshp sexport_=sexport
simport_=simport iexport_=iexport iimport_=iimport
```

Next, in order to enable us to identify specific sections of the social accounting matrix, we use *dynamic subsets* which is technical parlance for creating subsets of a already defined set. For instance, set `i` denotes a set for all institutions (i.e. households, government, etc.). In order to specifically identify all, say households in set `i`, we create dynamic subsets of the form:

```
set        h(i)        Private (household) institutions /
                hh1     Households LT10k    (10001)
                hh2     Households 10-15k   (10002)
                hh3     Households 15-25k   (10003)
                hh4     Households 25-35    (10004)
                hh5     Households 35-50k   (10005)
                hh6     Households 50-75k   (10006)
                hh7     Households 75-100k  (10007)
                hh8     Households 100-150k (10008)
```

```
                hh9     Households 150k+    (10009) /,
```

As will be apparent, this allows us to easily slice up the submatrices that form the aggregate social accounting matrix into benchmark parameters in a general equilibrium model. The next task takes our three dimensional arrays of data (remembering that IMPLAN data has additional transfer information) and converting it into the more standard two dimensional arrays more standard to AGE analysis and saving the additional transfer data into additional parameters. So, first, we specify our parameters of interest:

```
parameter       make(s,g)               Domestic industry make matrix
                use(g,s)               Industry use matrix
                iuse(g,i)               Institutional use matrix
                fd(f,s)                       Industry use of factors
...
```

Once specified, we then identify the appropriate elements in each three dimensional array of data. This looks like:

```
make(s,g)         = make_(s,"cmke",g);  make_(s,"cmke",g) = 0;
abort$card(make_) "Unexpected nonzeros here:", make_;
use(g,s)          = use_(g,"cuse",s);   use_(g,"cuse",s) = 0;
abort$card(use_)  "Unexpected nonzeros here:", use_;
iuse(g,i)         = iuse_(g,"cuse",i);  iuse_(g,"cuse",i) = 0;
abort$card(iuse_) "Unexpected nonzeros here:", iuse_;
fd(f,s)           = fd_(f,"frpt",s);    fd_(f,"frpt",s) = 0;
abort$card(fd_) "Unexpected nonzeros here:", fd_;
```

The above identifies each element of the three dimensional array that is needed for future analysis. For instance, in the first line, ''`cmke`'' refers to "commodity make" which is the relevant transfer information for our purposes. Additionally, ''`cuse`'' refers to "commodity use" and ''`frpt`'' refers to "factor receipts" (which are all located in Appendix A). The `abort$` commands are a simple failsafe that indicates if any nonzero elements are detected in the three dimensional arrays which are unexpected. As is consistently observed throughout this note, many verification procedures like this exist in the code.

Once the relevant parameters have been formulated, a series of algebraic calculations are undergone to ensure that accounting identities still hold after partitioning the social accounting matrix. We label these calculations as zero-profit, market clearance and income balance identities. For instance, the zero-profit condition is specified as:

```
parameter        bmkchk        Handshake on benchmark consistency;

*        Profitability of sectoral production:

bmkchk(s,"profit") = round(sum(g, make(s,g) - use(g,s))
                + sum((g,trd), sexport(s,g,trd)-simport(trd,g,s))
                - sum(f, fd(f,s)), 6);
```

In more readable algebra, letting $make_{s,g}$ be the make matrix, $use_{g,s}$ be the use matrix, $sexport_{s,g,trd}$ be the sectoral exports, $simport_{trd,g,s}$ be sectoral imports and $fd_{f,s}$ be factor demands, this is specified as:

$$\sum_{g}\left(make_{s,g}+\sum_{trd}sexport_{s,g,trd}\right)=\sum_{g}\left(use_{g,s}+\sum_{trd}simport_{trd,g,s}\right)+\sum_{f}fd_{f,s},\quad\forall s$$

This simply states that the amount of output produced and exported by sector s must equal the value of inputs used in production. Similar relationships exist for goods, factors and trade accounts. For the

sake of brevity, we leave these balancing checks for the perusal of the modeler. Such checks are later more explicitly described. However, as will be used later on, let $iexport_{i,g,trd}$ be institutional exports, $iimport_{trd,g,k}$ be institutional imports, $imake_{i,g}$ be the institutional make matrix, and $iuse_{g,i}$ be the institutional use matrix.

Once accounting identities have been verified given our initial partitioning of the social accounting matrix, we divide the data into smaller subsections to be used as benchmark parameters in CGE analysis. Consider Table 3 for an overview of benchmark data. As a matter of convenience, we specify the set for trade, `trd` as simply `t` for the remainder of this note.

Table 3: Benchmark Data

GAMS PARAMETER	SYMBOL	DESCRIPTION
`s0(g)`	$\overline{s}_g$	Aggregate output
`a0(g)`	$\overline{a}_g$	Aggregate domestic absorption
`m0(t,g)`	$\overline{m}_{tg}$	Aggregate foreign and domestic imports
`md0(g,*)`	$\overline{md}_{g.}$	Sectoral and institutional import demand
`x0(t,g)`	$\overline{x}_{tg}$	Aggregate exports to foreign or domestic markets
`d0(g)`	$\overline{d}_g$	Local supply
`y0(*,g)`	$\overline{y}_{.g}$	Sectoral and institutional supply
`dd0(g,*)`	$\overline{dd}_{g.}$	Sectoral and institutional domestic demand
`fd0(f,s)`	$\overline{fd}_{fs}$	Sectoral factor demand
`fs0(f,s)`	$\overline{fs}_{fs}$	Sectoral factor supply
`br0(*,*)`	$\overline{br}_{..}$	Bilateral transfers
`bt0(i)`	$\overline{bt}_i$	Aggregate bilateral transfers
`fe0(i,f)`	$\overline{fe}_{if}$	Factor endowments
`fm0(t,f)`	$\overline{fm}_{tf}$	Factor imports
`fx0(t,f)`	$\overline{fx}_{tf}$	Factor exports
`nxe0`	$\overline{nxe}$	National exchange endowment
`fxe0`	$\overline{fxe}$	Foreign exchange endowment

Note, the $(.)$ here denotes the use of either the i (institutional) or s (sectoral) subscript.

In order to formulate these benchmark parameters, the following calculations are performed:

```
x0(g,trd)  = sum(s,sexport(s,g,trd)) + sum(i,iexport(i,g,trd));
d0(g)  = sum(s,make(s,g)) + sum(i,imake(i,g));
s0(g) = d0(g) + xn0(g) + x0(g);
a0(g)    = d0(g) + mn0(g);
y0(s,g)  = make(s,g)  + sum(trd, sexport(s,g,trd));
...
```

We've suppressed quite a few lines of GAMS code for each calculation simply because such relationships can be easily identified and rationalized with a quick browse through the associated code. However, an algebraic description for the above is given for clarifying thoughts. First, aggregate exports to both the foreign and domestic markets are formulated as:

$$\overline{x}_{gt} = \sum_s sexport_{sgt} + \sum_i iexport_{igt}, \quad \forall g, t$$

This simply stipulates that total exports are comprised of both sector and institutional level exports to foreign and national markets. Next, the local supply of goods is formulated as:

$$\overline{d}_g = \sum_s make_{sg} + \sum_i imake_{ig}, \quad \forall g$$

which describes the total amount of commodity g made by sectors and institutions (given by the make matrices). Aggregate output is represented as:

$$\overline{s}_g = \overline{d}_g + \sum_t \overline{x}_{gt}, \quad \forall g$$

meaning that total output is comprised of local supply and exports to foreign and national markets. It is important to distinguish here the difference between local and national markets. For instance, if we have data on a single county, say Adams county, Wisconsin, the local supply concerns only Adams county while the national (sometimes referred to as domestic) supply concerns the rest of the United States. Next, aggregated domestic absorption is formulated as:

$$\overline{a}_g = \overline{d}_g + \overline{m}_{'dtrd',g}, \quad \forall g$$

which specifies the amount of a good g produced locally or imported from national markets. The last parameter formulation described here is sectoral supply. Such is formulated as:

$$\overline{y}_{sg} = make_{sg} + \sum_t sexport_{sgt}, \quad \forall s, g$$

This simply adds the amount of good g produced by sector s with the total amount of such good exported by such sector. The other parameters are defined analogously via the GAMS code.

Given these parameters, we again perform zero-profit, market clearance and income balance checks to make sure the data was properly divided and inherent data accounting identities still hold. As has been consistent throughout this text thus far, we only introduce and explain explicitly a fragment of the GAMS code for brevity. In the code, this looks like:

```
parameter        mdlchk        Cross check on consistency of model equilibrium;

*        Profitability of sectoral production:

mdlchk(s,"profit") =  sum(g, y0(s,g)- dd0(g,s) - md0(g,s)) - sum(f,
fd0(f,s)-fs0(f,s));

*        Demand and supply in the market for domestic goods:

mdlchk(g,"market") = sum(s,y0(s,g)) + sum(i,y0(i,g)) - x0(g) - xn0(g) + mn0(g)
- sum(s,dd0(g,s)) - sum(i,dd0(g,i));

*        Factor markets:

mdlchk(f,"market") = sum(i,fe0(i,f)) + sum(trd,fm0(trd,f)-fx0(trd,f)) -
sum(s,fd0(f,s)-fs0(f,s));
...
```

As already described above when concerned with submatrices such as the make and use arrays, these consistency checks are crucial for continuing to ensure that our benchmark equilibrium still exists. Con-

sider first the zero profit conditions inherent in an economic equilibrium.

$$\sum_g \overline{y}_{sg} + \sum_f \overline{fs}_{fs} = \sum_g \left(\overline{md}_{gs} + \overline{dd}_{gs} \right) + \sum_f \overline{fd}_{fs}, \quad \forall s$$

Such condition stipulates that the value of sectoral supply of goods and factors must equal the value of goods and factors used in the production for sector s's output. Next, we have a market clearance condition for domestic goods. This is:

$$\sum_s \overline{y}_{sg} + \sum_i \overline{y}_{ig} + \overline{m}_{'dtrd',g} = \sum_t \overline{x}_{tg} + \sum_s \overline{dd}_{gs} + \sum_i \overline{dd}_{is} \quad \forall g$$

I.e. in the market for domestic goods g, goods supplied by all sectors and institutions, as well as those imported from the national market must equal the amount exported out of the region as well as those demanded for domestic use by sectors and institutions (hence these market clearance conditions simply equate supply with demand). Note, in the above notation, $\overline{m}_{'dtrd'g}$ is used to represent only the imports coming from the domestic market. Next, we have similar conditions for factor markets:

$$\sum_i \overline{fe}_{if} + \sum_t \overline{fm}_{tf} + \sum_s \overline{fs}_{fs} = \sum_t \overline{fx}_{tf} + \sum_s \overline{fd}_{fs} \quad \forall f$$

Here, the supply of factor f in the region include total factor endowments over all institutions, factor imports and total factor supply by sectors must equal total factor demand and exports. Again, this condition reflects the market clearance idea that is inherent in the predefined SAM accounting identities.

As is quite standard in the realm of input output data reconciliation, a few imbalances were detected. Thus, a matrix balancing optimization routine is employed to enforce key accounting identities while minimizing the percent change in the data. In such routine, we allow certain parameters described above to be *variables* which are denoted in uppercase. By letting a few parameters vary just slightly, key identities can be enforced. The optimization routine can be found in the GAMS code where:

```
nonnegative
variables        DD0_, MD0_, Y0_, X0_, D0_, XN0_, FD0_, FS0_, FE0_,
                 MN0_;

equations        objdef, profit, gmarket, fmarket, trademarket, income,
                 domestic;

parameter        zeropenalty        Penalty factor for introduction of
                                 nonzeros/1000/;

objdef.. OBJ =e= sum((g,s)$dd0(g,s), abs(dd0(g,s))*sqr(DD0_(g,s)/dd0(g,s)-1)) +
                 sum((g,s)$md0(g,s), abs(md0(g,s))*sqr(MD0_(g,s)/md0(g,s)-1)) +
                 sum((f,s)$fd0(f,s), abs(fd0(f,s))*sqr(FD0_(f,s)/fd0(f,s)-1)) +
                 sum((f,s)$fs0(f,s), abs(fs0(f,s))*sqr(FS0_(f,s)/fs0(f,s)-1)) +
                 sum((f,i)$fe0(i,f), abs(fe0(i,f))*sqr(FE0_(i,f)/fe0(i,f)-1)) +
                 sum((g,i)$dd0(g,i), abs(dd0(g,i))*sqr(DD0_(g,i)/dd0(g,i)-1)) +
                 sum((g,i)$md0(g,i), abs(md0(g,i))*sqr(MD0_(g,i)/md0(g,i)-1)) +
                 sum(g$xn0(g),             abs(xn0(g))   *sqr(XN0_(g)/xn0(g)-1)) +
                 sum(g$x0(g),             abs(x0(g))   *sqr(X0_(g)/x0(g)-1)) +
                 sum(g$d0(g),             abs(d0(g))   *sqr(D0_(g)/d0(g)-1)) +
                 sum(g$mn0(g),             abs(mn0(g))   *sqr(MN0_(g)/mn0(g)-1)) +
                 sum((i,g)$y0(i,g),  abs(y0(i,g)) *sqr(Y0_(i,g)/y0(i,g)-1)) +
         zeropenalty *(
                 sum((g,s)$(dd0(g,s)=0), DD0_(g,s)) +
                 sum((g,s)$(md0(g,s)=0), MD0_(g,s)) +
                 sum((f,s)$(fd0(f,s)=0), FD0_(f,s)) +
```

```
                    sum((f,s)$(fs0(f,s)=0), FS0_(f,s)) +
                    sum((i,f)$(fe0(i,f)=0), FE0_(i,f)) +
                    sum((g,i)$(dd0(g,i)=0), DD0_(g,i)) +
                    sum((g,i)$(md0(g,i)=0), MD0_(g,i)) +
                    sum(g$(x0(g)=0),        X0_(g))    +
                    sum(g$(d0(g)=0),        D0_(g))    +
                    sum(g$(mn0(g)=0),        MN0_(g))   +
                    sum((i,g)$(y0(i,g)=0),  Y0_(i,g)) );

profit(s)..        sum(g, y0(s,g)) =e= sum(g, DD0_(g,s) + MD0_(g,s)) + sum(f,
                                       FD0_(f,s)-FS0_(f,s));
...
```

In the above, a few constraints were left out. They are defined in algebraic notation explicitly below. In order to minimize the amount of space needed to formulate the above, allow the objective function to be written in aggregate terms using A_{sg} as the variable and $\bar{a}_{sg}$ the benchmark parameter. Moreover, let Φ_{sg} denote all (s,g) pairs that are nonzero and Φ^c_{sg} denote all pairs that are zero in the benchmark. Pick γ to be some large penalty scalar. The above can be formulated as:

$$
\begin{aligned}
\min_{A_{sg}} \quad & \sum_{\Phi_{sg}} |a_{sg}| \left(\frac{A_{sg}}{a_{sg}} - 1 \right)^2 + \gamma \sum_{\Phi^c_{sg}} A_{sg} \\
\text{s.t} \quad & \sum_g \overline{y}_{sg} = \sum_g \left(DD_{gs} + MD_{gs} \right) + \sum_f \left(FD_{fs} - FS_{fs} \right) \quad \forall s \\
& D_g + \sum_t X_{tg} = \sum_s \overline{y}_{sg} + \sum_i Y_{ig} \quad \forall g \\
& D_g + MN_g = \sum_s DD_{gs} + \sum_i DD_{is} \quad \forall g \\
& \sum_i FE_{if} + \sum_t \left(\overline{fm}_{tf} - \overline{fx}_{tf} \right) = \sum_s \left(FD_{fs} - FS_{fs} \right) \quad \forall f \\
& \sum_{tg} X_{tg} + \sum_{it} \left(\overline{br}_{ti} - \overline{br}_{it} \right) + \sum_{ft} \left(\overline{fx}_{tf} - \overline{fm}_{tf} \right) = \sum_g \left(MN_g + \sum_s MD_{gs} + \sum_i MD_{gi} \right) \\
& \sum_g Y_{ig} + \sum_f FE_{if} + \sum_{i'} \left(\overline{br}_{ii'} - \overline{br}_{i'i} \right) + \sum_t \left(\overline{br}_{ti} - \overline{br}_{it} \right) = \sum_g \left(DD_{gi} + MD_{gi} \right) \quad \forall i
\end{aligned}
$$

The above algebraically characterizes our balancing routine. The objective function seeks to minimize the percent difference between variable values and parameters by a least squares formulation. γ represents a zero penalty. Thus, all elements of a given variable that is initially zero in the benchmark remain zero after optimally solving the program. This provides us a mechanism for inducing sparsity in the social accounting matrix (thus easing any future computational burdens). The constraints represent our zero-profit, market clearance and income balance constraints previously defined above. The difference now concerns replacing any benchmark parameter subject to change with its associated variables in each equation.

Once specifying this routine, the program then seeks to calibrate the data such that all accounting identities are satisfied. Once the program solves and initial benchmark figures are replaced by solutions to the nonlinear program, we apply a filter to the data set which drops any values that are less than 0.1% of the average value for such parameter. The code looks like:

```
trace("x0","ave") = sum((g), abs(x0(g)))/sum((g)$x0(g),1);
trace("d0","ave") = sum((g), abs(d0(g)))/sum((g)$d0(g),1);
trace("y0","ave") = sum((s,g), abs(y0(s,g)))/sum((s,g)$y0(s,g),1);
```

```
...
parameter        zerotol/4/;

x0(g)$(round(x0(g)/trace("x0","ave"),zerotol)=0) = 0;
d0(g)$(round(d0(g)/trace("d0","ave"),zerotol)=0) = 0;
y0(s,g)$(round(y0(s,g)/trace("y0","ave"),zerotol)=0) = 0;
```

As is readily apparent above, the average values for each parameter are calculated. The code then, using the conditional command `$` allows us to pick all elements of the parameter less than 0.1% of the average value and set them to zero. Again, this is a convenient method for inducing sparsity. After dropping such values, the calibration procedure is solved again.

In order to properly assess if the data has been calibrated correctly, an accounting general equilibrium model was constructed. In the build stream distribution, such model is found under the build directory named `mgemodel.gms`. The MPSGE code begin with:

```
$ontext
$model:implan

$sectors:
        Y(s)$sy(s)                      !       Sectoral production
        X(g)$s0(g)                      !       Export
        A(g)$a0(g)                      !       Demand for domestic goods
        M(g)$m0(g)                      !       Foreign import

$commodities:
        PD(g)$d0(g)         !       Local goods market
        PY(g)$s0(g)         !       Sectoral output
        PA(g)$a0(g)         !       Domestic goods market
...
```

Obviously, the entire model isn't included in the above snippet of code. The accounting model's variables are represented in Table 4.

The market flows of such model can be nicely represented in Figure 1. For more on the model specifics, see Appendix B. For the purpose of introducing the build stream, understanding variable names and market flows suffices for continuing the discussion. The idea of the accounting model is simply to diagnose any calibration issues following our matrix balancing routine. Once solved, the filtered data set is saved in a new `*.gdx` file:

```
execute_unload '..\temp\gdx\%ds%_filtered.gdx', i, f, s, g, d0, bt0,
        y0, dd0, md0, fd0, fs0, a0, mn0, d0, x0, xn0, br0, fe0, fm0, fx0, nxe0,
        fxe0;
```

4 The Social Accounting Matrix

At this point in the build stream process, we've properly calibrated the data set such that benchmark parameters represent a benchmark equilibrium. The next program, `samgen.gms`, creates a two dimensional social accounting matrix which can be read in directly to GAMS for general equilibrium analysis. Also, for interest in understanding the benchmark data, an excel file is created using this new SAM framework constructed for analysis.

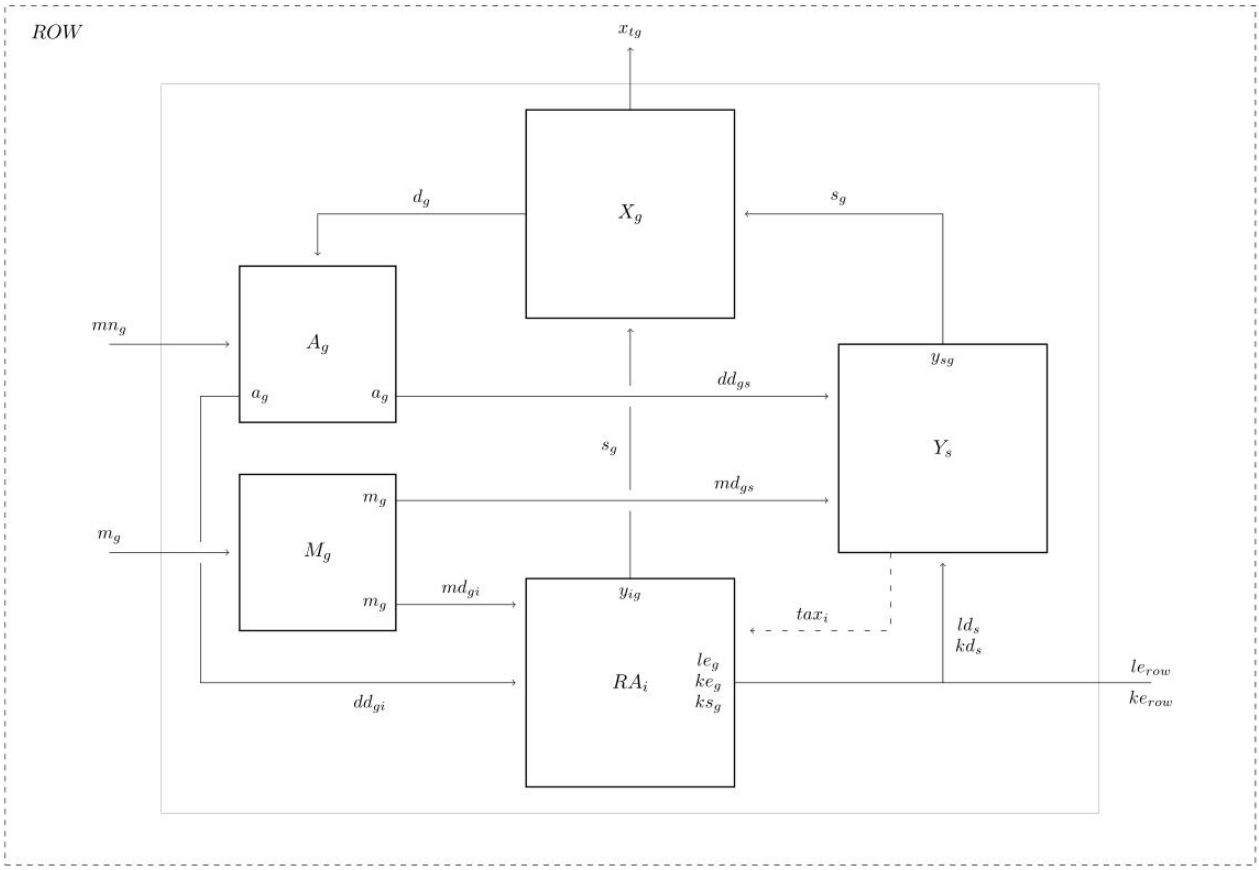
In this program, we construct a SAM using the logic from the accounting general equilibrium model outlined in Appendix B. First, we create sets for each element of the row/columns of this new SAM based on variables (both activity levels and prices) from the template CGE model. In the code, this looks like:

Table 4: GE Variables

VARIABLE	DESCRIPTION
Y_s	**Production** Production activities for the 440 sectors in the IMPLAN database
X_g	**Armington supply (foreign - national - local)** Local supply to foreign, national, or local markets
A_g	**Armington demand (national - local)** Demand for domestic goods
M_g	**Import supply** Imports supplied by foreign markets
RA_i	**Representative Agent**: Consumer type for each institution
ROW	**Rest of World**: Aggregate rest of world account
PY_g	**Output markets** Markets for commodities
PD_g	**Local markets** Local markets for commodities
PM_g	**Foreign imports** Imports used as final demand or inputs of production
PA_g	**Domestic composite** Domestic goods used as final demand or inputs of production
PF_f	**Factors of Production**: Factors as specified in the social accounting matrix
PT_i	**Bilateral transer market** Prices for transfers
PFX	**Foreign exchange**: Price for goods from the rest of world
PNX	**Domestic exchange**: Prices for goods from domestic market

```
set r Rows and columns in the SAM /

        1*440              Production (Y),
        1001*1440        Armington supply (foreign - national - local) (X),
        2001*2440        Armington demand (national - local) (A),
        3001*3440        Import supply (M),

        4001*4440        Output markets (PY),
        5001*5440        Local markets (PD),
        6001*6440        Foreign import (PM),
        7001*7440        Domestic composite (PA),

        8001               Employee Compensation
        8002               Proprietary Income
        8003               Other Property Income
        8004               Indirect Business Taxes

        10001*10009        Households
```

Figure 1: Intermediate Demands




```
        11001               Federal Government NonDefense
...
```

Such can be nicely summarized in Table 5.

Once the sets are explicitly defined, the social accounting matrix is constructed element by element via the `loop` command. Examples include:

```
set        industry1(r)        Y        /1/
        supply1(r)        X        /1001/
        absorb1(r)        A        /2001/
        import1(r)        M        /3001/
        output1(r)        PY        /4001/
        foreign1(r)        PM        /6001/
        armington1(r)        PA        /7001/;
...

*        PROD:Y

loop(output1(r),    loop(g,        sam(s,r+(g.val-1)) = y0(s,g);););
loop(armington1(r), loop(g,        sam(r+(g.val-1),s) = dd0(g,s);););
loop(foreign1(r),   loop(g,        sam(r+(g.val-1),s) = md0(g,s);););
```

The above snippet of code indicates two things. First, singleton sets are specified in order to tell GAMS

Table 5: New SAM Elements

HEADER	ELEMENTS	DESCRIPTION
Y		**Production**
	$1 * 440$	Production activities for the 440 sectors in the IMPLAN database
X		**Armington supply (foreign - national - local)**
	$1001 * 1440$	Local supply to foreign, national, or local markets
A		**Armington demand (national - local)**
	$2001 * 2440$	Demand for domestic goods
M		**Import supply**
	$3001 * 3440$	Imports supplied by foreign markets
PY		**Output markets**
	$4001 * 4440$	Markets for commodities
PD		**Local markets**
	$5001 * 5440$	Local markets for commodities
PM		**Foreign imports**
	$6001 * 6440$	Imports used as final demand or inputs of production
PA		**Domestic composite**
	$7001 * 7440$	Domestic goods used as final demand or inputs of production
PF		**Factors of Production**:
	8001	Employee Compensation
	8002	Proprietary Income
	8003	Other Property Income
	8004	Indirect Business Taxes
i		**Institutions**:
		Private (household) institutions:
	10001	Households less than 10k
	10002	Households 10-15k
	10003	Households 15-25k
	10004	Households 25-35k
	10005	Households 35-50k
	10006	Households 50-75k
	10007	Households 75-100k
	10008	Households 100-150k
	10009	Households 150k+
		Public (government) institutions:
	11001	Federal Government NonDefense
	11002	Federal Government Defense
	11003	Federal Government Investment
	12001	State Local Govt NonEducation
	12002	State Local Govt Education
	12003	State Local Govt Investment
		Corporate institutions:
	13001	Enterprises (Corporations)
	14001	Gross Private Fixed Investment (Capital)
	14002	Inventory Additions Deletions
t		**Trade**:
	25001	Foreign Trade
	28001	Domestic Trade

where benchmark data should be placed in the two dimensional SAM. Then looping over these singleton sets, benchmark parameters are read into the social accounting matrix. Perhaps the best way to illustrate what is going on is to simply give the overview of what this new social accounting matrix looks like.

Table 6 explicitly shows what we mean by "dropping" benchmark parameters into the SAM via the `loop` command. Notably, the row and column sums of such matrix are precisely what we referred to as zero-profit, market clearance and income balance conditions mentioned above.

Notice that the bottom left quadrant of this matrix is shaded in gray. Due to the additional information that IMPLAN provides on transfers between institution types, the gray cells indicate additional information on such transfers for these parameters. This data is stored in a satellite table and can be used in the model if necessary. In the code, this would look like:

```
parameter        sattelite(r,rr,tn)        Sattelite disaggregation of SAM transfers;

loop(imap(i,r),

*        Transfers between domestic agents:

  loop(iimap(ii,rr),
        sam(r,rr) = sum(t,trnsfer(i,t,ii));
        loop(mapt(t,tn),sattelite(r,rr,tn) = trnsfer(i,t,ii));  );
```

Again, we use the same programming techniques to construct the satellite tables, though we restrict our attention to only transfer data. Once the tables have been constructed, a series of balancing checks are performed:

```
set        sambalcol /"Row Sum","Column Sum","Difference"/;
parameter        sambalance        Cross check on microconsistency of the social
accounting matrix;
sambalance(r,"Row Sum") = sum(rr,sam(r,rr));
sambalance(r,"Column Sum") = sum(rr,sam(rr,r));
sambalance(r,"Difference") = sambalance(r,"Row Sum") - sambalance(r,"Column Sum");
sambalance(r,sambalcol)$(not round(sambalance(r,"Difference"),6)) = 0;
display sambalance;
```

This check serves to illustrate that no errors were made when mapping benchmark parameters to the social accounting matrix. Thereafter, the constructed matrices are translated into readable excel files:

```
$onecho >..\data\xlsx\gdxxrw.rsp
set=rdef rng="Accounts!a1" rdim=3 cdim=0 values=string
set=tnm  rng="Accounts!e1" rdim=1 cdim=0 values=string
par=sam  rng="SAM!a1" cdim=0
par=sattelite  rng="Sattelite!a1" cdim=0
$offecho
execute_unload '..\temp\gdx\%ds%_sam.gdx', rdef, tnm, sam, sattelite;
execute 'gdxxrw i=..\temp\gdx\%ds%_sam.gdx o=..\data\xlsx\%ds%_sam.xlsx
                                %@..\data\xlsx\gdxxrw.rsp';
```

The `$onecho` allows us to write a little script to convert the data into a `*.gdx` file and then use `gdxxrw.exe` to convert the `gdx` file to an excel file. Note that `rdim` and `cdim` indicate row and column dimension of the data, `par` denotes the parameter of interest and `rng` allows us to specify which tab in excel to place the data. The resulting excel file will be named `%ds%_sam.xlsx` and will be located in the xlsx data directory.

Table 6: Regional Social Accounting Matrix: New Framework

		Production	Exports	Aggregate Composite	Output Markets	Local Market	Import Market	Domestic Composite	Factors	Institutions	Trade
		Y	X	A	PY	PD	PM	PA	PF	i	t
Production	Y				$\overline{y}_{sg}$				$\overline{fs}_{fs}$		
Exports	X					$\overline{d}_g$					$\overline{x}_{tg}$
Aggregate Composite	A							$\overline{a}_g$			
Output Markets	PY		$\overline{s}_g$								
Local Market	PD			$\overline{d}_g$							
Import Market	PM	$\overline{md}_{gs}$								$\overline{md}_{gi}$	
Domestic Composite	PA	$\overline{dd}_{gs}$								$\overline{dd}_{gi}$	
Factors	PF	$\overline{fd}_{fs}$									$\overline{fx}_{tf}$
Institutions	i				$\overline{y}_{ig}$				$\overline{fe}_{if}$	$\overline{br}_{ii'}$	$\overline{br}_{it}$
Trade	t			$\overline{mn}_{tg}$			$\overline{m}_{tg}$		$\overline{fm}_{tf}$	$\overline{br}_{ti}$	

5 Partitioning and Benchmark Consistency

The final step that remains in the build stream concerns taking the previously outputted excel file, reading it back into GAMS, defining benchmark parameters from such excel file, and verifying via the CGE accounting model that the benchmark data still represents an equilibrium. The program begins by defining sets as in the `samgen.gms` program. I.e.

```
set r Rows and columns in the SAM /
        1*440               Production (Y),
        1001*1440        Armington supply (foreign - national - local) (X),
        2001*2440        Armington demand (national - local) (A),
        3001*3440        Import supply (M),

        4001*4440        Output markets (PY),
        5001*5440        Local markets (PD),
        6001*6440        Foreign import (PM),
        7001*7440        Domestic composite (PA),
...
```

Once all sets have been declared, the excel file `%ds%_sam.xlsx` is converted to a `gdx` file with `gdxxrw.exe` and then read into the program. Once the data is read in, we perform checks verifying that microconsistency still holds (eliminating the risk that errors occurred when reading the data).

```
set               chkcol /Rowsum,ColSum,Tolerance/;
parameter        samchk               Cross check on SAM;
loop(rdef(r,acct,u),
        samchk(r,acct,u,"RowSum") = sum(rr, sam(r,rr));
        samchk(r,acct,u,"ColSum") = sum(rr, sam(rr,r));
        samchk(r,acct,u,"Tolerance") = sum(rr, sam(r,rr)-sam(rr,r)););
samchk(r,acct,u,chkcol)$(not round(samchk(r,acct,u,"Tolerance"),3)) = 0;
option samchk:3:3:1;
display samchk;
```

Once we've verified row and column sums match, we proceed to "undo" the looping mechanism described in the previous program. That is, we forulate benchmark parameters used in the accounting CGE model based on the aggregate SAM. This looks like:

```
parameter
        s0(g)               Aggregate output,
        a0(g)               Aggregate domestic absorption,
        m0(g)               Aggregate foreign imports,
        d0(g)               Local supply,
        x0(g)               Foreign exports,
...
loop((rdef(r,"y",s), cdef(rr,"py",g)),   y0(s,g)  = sam(r,rr);   sam(r,rr)=0;);
loop((cdef(rr,"y",s),rdef(r,"pa",g)),    dd0(g,s) = sam(r,rr);   sam(r,rr)=0;);
loop((cdef(rr,"y",s),rdef(r,"pm",g)),    md0(g,s) = sam(r,rr);   sam(r,rr)=0;);
...
```

As is apparent, we declare our standard parameter names, then select from the aggregate SAM the appropriate data elements for each parameter. We've defined dynamic subsets `cdef` and `rdef` to easily select the appropriate data elements via the `loop` command. Once all parameter are created, we again verify that the standard accounting identities inherent in input-output data hold.

```
parameter        profit               Zero profit chk;
profit(s,"y") = sum(g, y0(s,g))
                - sum(g,dd0(g,s)+md0(g,s))
                - sum(f, fd0(f,s)-fs0(f,s));
```

```
profit(g,"a") = a0(g) - mn0(g) - d0(g);
profit(g,"x") = x0(g)+xn0(g)+d0(g)-s0(g);
display profit;
...
```

The above specifies our zero-profit check on the data. Finally, the last step is to verify that the benchmark data does indeed describe an equilibrium. Thus, it is read into the accounting CGE model described in Appendix B.

```
$include mgemodel

implan.iterlim = 0;
$include implan.gen
solve implan using mcp;
```

The above bit of code conveniently includes the MPSGE model in this program without explicitly defining the model within the program. `implan.iterlim=0` sets the iteration limit for the model "implan" to zero, thus allowing us to verify that benchmark equilibrium holds. Indeed, by setting the iteration limit to zero, the level values for each variable (benchmark data) are plugged into the model. Benchmark equilibrium replication is verified if the solutions to the model are unity and the residual term is insignificant.

If properly calibrated, the data is outputted into a final `gdx` data file. Moreover, we include a measure of data precision in the `gdx` file which represents the residual term.

```
parameter        tolerance        Data precision;
tolerance = implan.objval;

execute_unload '..\data\GAMSData\%ds%.gdx', i, f, s, g, d0, bt0,
        y0, dd0, md0, fd0, fs0, a0, mn0, d0, x0, xn0, br0, fe0, fm0, fx0, nxe0,
        fxe0, tolerance;
```

6 Using the Batch File

This section will be of primary interest to individuals that accept our build routine as is and are simply looking to extract from their IMPLAN data usable CGE benchmark parameters. Our batch file, `run.bat`, contains the means to automatically set up the necessary directory structure as well as reconcile all available IMPLAN datasets. Importantly, it should be noted that each dataset must be in its `*.gms` file format which can be directly outputted by the IMPLAN software.

The batch file begins with:

```
@echo off

set        data=..\data\
set        temp=..\temp\
set        GAMSData=..\data\GAMSData\

:        For a Unix script:

:        set        datadir=../data/
:        set        tempdir=../temp/

:        Create the requisite output directories:

if not exist %data%nul                mkdir %data%
if not exist %temp%nul                mkdir %temp%
```

```
if not exist %GAMSData%nul        mkdir %GAMSData%
if not exist %temp%gdx\nul        mkdir %temp%gdx
if not exist %temp%listing\nul        mkdir %temp%listing
if not exist %temp%restart\nul        mkdir %temp%restart
if not exist %data%\xlsx\nul        mkdir %data%xlsx
...
```

The default configuration for this batch file is for *Windows* machines. Later versions will contain Mac friendly parallels. The program begins by constructing data directories. The `set` commands allow us to dynamically assign directory paths for each category. If the modeler is interested in changing such paths, simply alter the path configuration in each `set` environment. Then, the `mkdir` commands makes each specified directory.

Next, we generate each data set with:

```
:        Generate datasets for a specific set of regions:

set datadir=%data%Wisconsin
set regions=
setlocal EnableDelayedExpansion
for %%f in (%datadir%\*.gms) do (set regions=!regions! %%~nf)

:        Alternatively could process all the states:

:set datadir=%data%States
:set regions=
:setlocal EnableDelayedExpansion
:for %%f in (%datadir%\*.gms) do (set regions=!regions! %%~nf)


call readimplan %regions%
call condense   %regions%
call samgen     %regions%
call partition  %regions%
```

Importantly, we have our root Wisconsin data directory located in `...\data\Wisconsin`. However, this must be changed to agree with your machine. For instance, if you have Illinois county data in `...\data\Illinois`, change the first line above to `set datadir=%data%Illinois`. Once the correct data directory has been entered in, the program automatically generates data sets for all data files in such directory (by calling on the four previously described programs for each region). Finally, a measure of tolerance for data set precision is stored using:

```
:tolerances
gdxmerge %GAMSData%*.gdx id=tolerance
echo.
echo        Resulting dataset precision:
echo.
gdxdump merged.gdx symb=tolerance
:end
```

References

IMPLAN-Group-LLC. (2014): "IMPLAN User's Guide v3," .

Mathiesen, L. (1985): "Computation of economic equilibria by a sequence of linear complementarity problems," *Mathematical Programming Study*, 23, 144–162.

Rausch, S., and T. F. Rutherford (2009): "Tools for building national economic models using state-level implan social accounts," *Unpublished manuscript*.

Rosenthal, R. E. (2004): "GAMS–a user's guide," .

Rutherford, T. F. (1995): "Extension of GAMS for complementarity problems arising in applied economics," *Journal of Economic Dynamics and Control*, 19(8), 1299–1324.

A New GAMS Set Identifiers

We first note that numbers are still used when referencing sectors. However, because sectors and commodities are identical in the data set, we simply use one number index to indicate either a good or a sector.

Table 7: New Set Identifiers

ACTIVITY	GAMS INDEX	DESCRIPTION
Goods and Sectors:		
	$1 * 440$	Production activities for the 440 sectors in the IMPLAN database
Factors of Production:		
	empl	Employee Compensation (5001)
	prop	Proprietary Income (6001)
	othp	Other Property Income (7001)
	btax	Indirect Business Taxes (8001)
Private (household) institutions:		
	hh1	Households LT10k (10001)
	hh2	Households 10-15k (10002)
	hh3	Households 15-25k (10003)
	hh4	Households 25-35k (10004)
	hh5	Households 35-50k (10005)
	hh6	Households 50-75k (10006)
	hh7	Households 75-100k (10007)
	hh8	Households 100-150k (10008)
	hh9	Households 150k+ (10009)
Public (government) institutions:		
	fnd	Federal Government NonDefense (11001)
	fdf	Federal Government Defense (11002)
	fin	Federal Government Investment (11003)
	sln	State Local Govt NonEducation (12001)
	sle	State Local Govt Education (12002)
	sin	State Local Govt Investment (12003)
Corporate institutions:		
	ent	Enterprises (Corporations) (13001)
	inv	Gross Private Fixed Investment (Capital) (14001)
	stk	Inventory Additions Deletions (14002)
Transfers:		
	cprf	Corporate Profits with IVA (15001)
	ncmp	Emp Comp (Wages/Salary w/o Soc Sec) (15002)
	ecmp	Employee Comp (Other Labor Income) (15003)
	prop	Proprietors Inc (w/o Soc Sec & CCA) (15004)
	rent	Rent with Capital Consumption Adj (15005)

ACTIVITY	GAMS INDEX	DESCRIPTION
	btrn	Business Transfers (15006)
	divd	Dividends (15007)
	nint	Interest (net - from industries) (15008)
	gint	Interest (gross) (15009)
	trns	Transfers (15010)
	srpl	Surplus or deficit (15011)
	save	Savings (surplus) not use (15012)
	wage	Wage accruals less surplus (15013)
	sstw	Social ins tax - employee Contribution (15014)
	sstf	Social ins tax - employer Contribution (15015)
	sgov	Surplus-subsidy - govt enterprises (15016)
	excs	Indirect bus tax: Excise Taxes (15017)
	duty	Indirect bus tax: Custom Duty (15018)
	nont	Indirect bus tax: Fed NonTaxes (15019)
	stax	Indirect bus tax: Sales Tax (15020)
	bptx	Indirect bus tax: Property Tax (15021)
	bmvt	Indirect bus tax: Motor Vehicle Lic (15022)
	sevt	Indirect bus tax: Severance Tax (15023)
	otax	Indirect bus tax: Other Taxes (15024)
	slnt	Indirect bus tax: S/L NonTaxes (15025)
	ctax	Corporate profits tax (15026)
	pitx	Personal Tax: Income Tax (15027)
	egtx	Personal Tax: Estate and Gift Tax (15028)
	fees	Personal Tax: NonTaxes (Fines- Fees (15029)
	pmvt	Personal Tax: Motor Vehicle License (15030)
	pptx	Personal Tax: Property Taxes (15031)
	fish	Personal Tax: Other Tax (Fish/Hunt) (15032)
	capc	Capital consumption allowance (15033)
	retp	Retained profits (profits w/IVA&CCA (15034)
	disc	NIPA statistical discrepency (15035)
	fint	Interest (net -from RoW) (15036)
	fact	Factor trade (15037)
	radj	Adjustment to retained earnings (15038)
	cuse	Commodity Use (15050)
	ctrd	Commodity Trade (15051)
	cmke	Commodity Make (15052)
	frpt	Factor Receipts (15053)
	ftrn	Foreign Commodity Transshipments (15054)
	iuse	Industry Use (15055)
	itrd	Industry Trade (15056)
Trade:		
	ftrd	Foreign Trade (25001)
	dtrd	Domestic Trade (28001)

B Accounting Model

B.1 The Primal Formulation

The benchmark identities presented in the previous section indicate the market clearance, zero profit, and income balance conditions but do not, however, characterize the behavior of agents in the model. In the competitive equilibrium setting, the standard assumption of optimizing atomistic agents applies for both producers and consumers. For sector Y_s, we characterize input choices as though they arose from minimizing unit costs:

$$\begin{aligned} \min_{dd_{gs}, md_{gs}, fd_{fs}} \quad & c_s^A + c_s^M + c_s^F \\ \text{s.t.} \quad & c_s^A = \sum_g PA_g dd_{gs} \\ & c_s^M = \sum_g PM_g md_{gs} \\ & c_s^F = \sum_f PF_f fd_{fs} \\ & F_s(dd_{gs}, md_{gs}, fd_{fs}) = Y_s \end{aligned}$$

FIGURE 2: PRODUCTION FUNCTION IN MPSGE SYNTAX

```
$prod:Y(s)$sy(s)  t:0 s:0  g.tl:4  va:1
        o:PY(g)         q:y0(s,g)
        i:PA(g)         q:dd0(g,s)      g.tl:
        i:PM(g)         q:md0(g,s)      g.tl:
        i:PF(f)         q:fd0(f,s)      va:
        o:PF(f)         q:fs0(f,s)
```

Note here that we denote decision variables as the same benchmark parameter symbol excluding the overhead bar. The production function F is described by a nested constant-elasticity-of-substitution (CES) form which is shown in MPSGE syntax as indicated by *i:* fields in Figure 2. The supply of output Y_s is portrayed as arising from the following profit-maximization problem:

$$\begin{aligned} \max_{y_{sg}, (fs_{fs})} \quad & PY_g y_{sg} + PF_f(fs_{fs}) \\ \text{s.t.} \quad & Y_s = \Gamma_s(y_{sg}, fs_{fs}) \end{aligned}$$

The production function $\Gamma()$ is described by a constant-elasticity-of-transformation (CET) form which is shown in MPSGE syntax as indicated by *o:* fields in Figure 2. The *g.tl* and *va* fields denote differing elasticities of substitution at the various nests of the production function (i.e. input imports from domestic vs. foreign markets have a different elasticity from factor markets).

The choice among imports from different trading partners is based on Armington's idea of regionally differentiated products. Here, we differentiate between goods produced locally (at price PD_g and goods produced in other US counties (at price PNX). This is reflected by the following cost minimization

problem:

$$\begin{aligned} \min_{mn_g, d_g} \quad & (PNX)mn_g + PD_g d_g \\ \text{s.t.} \quad & F_g^A(mn_g, d_g) = A_g \end{aligned}$$

Figure 3: Armington Aggregation in MPSGE syntax

```
$prod:A(g)$a0(g)  s:8
        o:PA(g)          q:a0(g)
        i:PNX            q:mn0(g)
        i:PD(g)          q:d0(g)
```

The import aggregation function portrayed by F_g^A above is described by the nested CES function shown in Figure 3.

Figure 4: Imports

```
$prod:M(g)$m0(g)
        o:PM(g)          q:m0(g)
        i:PFX            q:m0(g)
```

We then differentiate domestic demand for goods from foreign imports of goods. This is represented in Figure 4. As shown, we have a Leontief production function where the supply of foreign imports at price PM_g to the local market has input prices PFX. This is denoted in the following cost minimizing problem:

$$\begin{aligned} \min_{m_g} \quad & (PFX)m_g \\ \text{s.t.} \quad & \Gamma^M(m_g) = M_g \end{aligned}$$

Figure 5: Exports

```
$prod:X(g)$s0(g)  t:4
        o:PFX            q:x0(g)
        o:PNX            q:xn0(g)
        o:PD(g)          q:d0(g)
        i:PY(g)          q:s0(g)
```

Next, we characterize the exports market by Figure 5. Supply from local sources either are used locally (at price PD_g), exported to other counties (at price PNX) or exported to foreign markets (at price PFX). This market can be represented by a constant elasticity of transformation production function (as denoted

by the use of the *t:* field). I.e. we can define the problem as follows:

$$\max_{x_g, xn_g, d_g} \quad (PFX)x_g + (PNX)xn_g + PD_g d_g$$
$$\text{s.t.} \quad F^X(x_g, xn_g, d_g) = X_g$$

Figure 6: Representative Agents

```
$demand:RA(i)  s:1  g.tl:4
        d:PA(g)          q:dd0(g,i)      g.tl:
        d:PM(g)          q:md0(g,i)      g.tl:
        d:PT(i)          q:bt0(i)
        e:PY(g)          q:y0(i,g)
        e:PT(ii)         q:br0(ii,i)
        e:PNX            q:(sum(trd,br0(trd,i)))
        e:PF(f)          q:fe0(i,f)
```

Now considering the demand side of the model, representative agents maximize utility subject to their budget constraint. This is reflected in the MPSGE code in Figure 6. Here, we model demand by assuming Cobb-Douglas preferences amongst transfers, factors and goods (note that this is ascribed by the *s:* field). Moreover, the elasticity of substitution between foreign imports and domestic goods is 4. The associated optimization problem involved is:

$$\max_{dd_{gi}, md_{gi}, bt_i} U(dd_{gi}, md_{gi}, bt_i)$$
$$\text{s.t. } RA_i = \sum_g PY_g y_{ig} + \sum_{i'} PT_{i'} br_{ii'} + PNX(\sum_t br_{ti}) + \sum_f PF_f fe_{if}$$

Figure 7: Rest of World

```
$demand:ROW
        d:PFX
        e:PNX            q:(-sum((i,trd),br0(trd,i)))
        e:PT(i)          q:(sum(trd,br0(i,trd)))
        e:PF(f)          q:(sum(trd,fm0(trd,f)-fx0(trd,f)))
        e:PNX            q:nxe0
        e:PFX            q:fxe0
```

Finally, we must consider the rest of the world. Consider Figure 7. Similar to the case of the representative agent, Figure 7 can be represented as a utility maximization problem. Consider the following:

$$\max U$$
$$\text{s.t.} \quad ROW = PNX(-\sum_{it} br_{ti}) + \sum_{it} PT_i(\sum_t br_{it})$$
$$+ PNX(nxe) + PFX(fxe)$$

Note, it is important to realize that by setting reference quantities to our benchmark parameters, the

solutions to the model will be consistent with prices simply set to 1. In a sense, then, without introducing any sort of demand or supply side shock, this model serves as an accounting model for our benchmark values (and correct calibration). Thus, introducing a shock to the model allows us to analyze how relative prices change.

B.2 Equilibrium Conditions

We now define the general equilibrium of the model in a complementarity format. Rutherford (1995) and Mathiesen (1985) have shown that a complementary-based approach is convenient, robust, and efficient. A characteristic of economic models is that they naturally involve a complementary problem, i.e. given a function F: $\mathbb{R}^n \longrightarrow \mathbb{R}^n$, find $z \in \mathbb{R}^n$ such that $F(z) \geq 0$, $z \geq 0$, and $z^T F(z) = 0$.

An equilibrium in complementarity format is represented by a vector of activity levels, a non-negative vector of prices, and a non-negative vector of incomes such that:

1. no production activity makes a positive profit (zero profit conditions),
2. excess supply is non-negative for all goods and factors (market clearance conditions), and,
3. expenditure does not exceed income (budget constraints).

Zero-profit conditions exhibit complementary slackness with respect to associated activity levels, market clearing exhibit complementary slackness with respect to market prices, and budget constraint define income variables. Notably, formulating general equilibrium models in a complementarity format is useful for solving large models. Attempting to solve the equivalent large scale nonlinear program (those outlined above) for the model equilibrium proves quite difficult with many variables and equations.

B.3 GAMS/MCP Formulation

We follow the Harberger and Peronni approaches for solving the MCP formulation by initially setting all prices and activity levels to 1. Therefore, in the following calibrated form equations, benchmark prices are set to 1 and thus do not appear.

Let the following notation serve as the necessary elasticities in a CES framework. First the elasticity of substitution between local and national goods is defined as ($\sigma_g^{dn} = 8$). The Armington elasticity of substitution between domestic and foreign goods is given as ($\sigma_{g.}^{dm} = 4$). Furthermore, the elasticity of transformation used in the export market is defined as ($\eta_g^x = 4$).

B.4 Zero-Profit Conditions

Define the following as benchmark value shares, θ, benchmark value added, va_s, and benchmark demands, id_{gs} to be used in our CES formulation of the production market:

$$\begin{aligned}
\theta_{fs}^{va} &= \frac{\overline{fd}_{fs}}{\overline{va}_s} \\
\theta_{gs}^{m} &= \frac{\overline{md}_{gs}}{\overline{id}_{gs}} \\
\overline{va}_s &= \sum_f \overline{fd}_{fs} \\
\overline{id}_{gs} &= \overline{md}_{gs} + \overline{dd}_{gs}
\end{aligned}$$

The zero-profit condition for the production market (Y) stipulates that the cost of production must be equated with revenue. Thus, we define cost functions as follows:

$$\begin{aligned} CVA_s &= \prod_f PF_f^{\theta_{fs}^{va}} \\ PDY_{gs} &= \left(\theta_{gs}^m PM_g^{1-\sigma_{gs}^{dm}} + (1-\theta_{gs}^m) PA_g^{1-\sigma_{gs}^{dm}}\right)^{1/1-\sigma_{gs}^{dm}} \end{aligned}$$

CVA denotes the cost function for factors used in production (exhibiting Cobb-Douglas technologies) and PDY indicates the cost function for domestic and foreign goods used as inputs in production. Thus, the zero-profit condition for market Y_s can be defined as:

$$\overline{va}_s CVA_s + \sum_g (\overline{id}_{gs} PDY_{gs}) \geq \sum_g (PY_g \overline{y}_{sg}) + \sum_f (PF_f \overline{fs}_{fs})$$

Next, domestic absorption of inputs combines local goods and imports from other domestic regions. Let θ_g^n denote the domestic value share (note, domestic and national is used synonymously) where:

$$\theta_g^n = \frac{\overline{mn}_g}{(\overline{mn}_g + \overline{d}_g)}$$

We therefore can define the cost function for national and local goods and thus the zero profit condition as:

$$CA_g = \left(\theta_g^n PNX^{1-\sigma_g^{dn}} + (1-\theta_g^n) PD_g^{1-\sigma_g^{dn}}\right)^{1/1-\sigma_g^{dn}}$$

$$(\overline{mn}_g + \overline{d}_g) CA_g \geq \overline{a}_g PA_g$$

Because the import market simply has one input and one output with the same reference quantity, the zero-profit condition can be written as:

$$PFX = PM_g$$

For the export market, I can define the benchmark value shares as follows:

$$\begin{aligned} \alpha_g^x &= \frac{\overline{x}_g}{\overline{x}_g + \overline{xn}_g + \overline{d}_g} \\ \alpha_g^{nx} &= \frac{\overline{xn}_g}{\overline{x}_g + \overline{xn}_g + \overline{d}_g} \\ \alpha_g^d &= \frac{\overline{d}_g}{\overline{x}_g + \overline{xn}_g + \overline{d}_g} \end{aligned}$$

Where α_g^x denotes the foreign export share, α_g^{nx} denotes the national export share, and finally α_g^d denotes the local export share. We define the unit revenue as:

$$RX_g = \left(\alpha_g^x PFX^{1+\eta_g^x} + \alpha_g^{nx} PNX^{1+\eta_g^x} + \alpha_g^d PD_g^{1+\eta_g^x}\right)^{1/1+\eta_g^x}$$

And thus, the zero-profit condition for exports is:

$$\bar{s}_g PY_g = (\bar{x}_g + \overline{xn}_g + \bar{d}_g) RX_g$$

B.5 Market Clearance

The next necessary condition for an economic equilibrium requires that the supply of good g must be greater than or equal to the demand of good g for all g. Before stating the market clearance conditions, additional value shares and demand functions are needed. Let θ^d_{gi} denote the value share of final demand, θ^t_i denote the transfers share of expenditure, and θ^m_{gi} be the institutional import value share. Also, let the benchmark institutional demand be $\overline{id}_{gi}$. That is:

$$\overline{id}_{gi} = \overline{md}_{gi} + \overline{dd}_{gi}$$

$$\theta^m_{gi} = \frac{\overline{md}_{gi}}{\overline{id}_{gi}}$$

$$\theta^d_{gi} = \frac{\overline{id}_{gi}}{\overline{bt}_i + \sum_g \overline{id}_{gi}}$$

$$\theta^t_i = \frac{\overline{bt}_i}{\overline{bt}_i + \sum_g \overline{id}_{gi}}$$

Before defining the demand functions for imported and domestic goods and transfers, we first define the bottom level of the nested expenditure functions.

$$PDRA_{gi} = \left(\theta^m_{gi} PM_g^{1-\sigma^{dm}_{gi}} + (1 - \theta^m_{gi}) PA_g^{1-\sigma^{dm}_{gi}} \right)^{1/1-\sigma^{dm}_{gi}}$$

Therefore, the demand functions for imports, domestic goods, and transfers can be defined as (note, because the elasticity of substitution is 1, they are Cobb-Douglas):

$$MDRA_{gi} = \left(\frac{PDRA_{gi}}{PM_g} \right)^{\sigma^{dm}_{gi}} \theta^m_{gi} \theta^d_{gi} \left(\frac{RA_i}{PDRA_{gi}} \right)$$

$$DDRA_{gi} = \left(\frac{PDRA_{gi}}{PA_g} \right)^{\sigma^{dm}_{gi}} (1 - \theta^m_{gi}) \theta^d_{gi} \left(\frac{RA_i}{PDRA_{gi}} \right)$$

$$TD_i = \left(\frac{\theta^t_i RA_i}{PT_i} \right)$$

Now that the demand functions have been defined, the market clearance conditions necessary for a model equilibrium require that supply equals demand. For the local market:

$$\bar{d}_g X_g \left(\frac{PD_g}{RX_g} \right)^{\eta^x_g} = \bar{d}_g A_g \left(\frac{CA_g}{PD_g} \right)^{\sigma^{dn}_g}$$

That is, the supply of good, g to the local market must be equated with the demand for such good. Notably, this equation can be derived from taking the derivative of the respective cost function and multiplying that by the associated activity level in the market of interest. Similarly, for output markets, PY, we have:

$$\sum_{sy_s} Y_s \overline{y}_{sg} + \sum_i \overline{y}_{ig} = \overline{s}_g X_g$$

Market clearance for PA:

$$A_g \overline{a}_g = \sum_{sy_s} Y_s \overline{dd}_{gs} \left(\frac{PDY_{gs}}{PA_g} \right)^{\sigma_{gs}^{dm}} + \sum_i DDRA_{gi}$$

Market clearance for PM:

$$M_g \overline{m}_g = \sum_{sy_s} Y_s \overline{md}_{gs} \left(\frac{PDY_{gs}}{PM_g} \right)^{\sigma_{gs}^{dm}} + \sum_i MDRA_{gi}$$

Market clearance for PT:

$$\sum_{i'} \overline{br}_{ii'} + \sum_t \overline{br}_{it} = TD_i$$

Market clearance for PF:

$$\sum_{sy_s} Y_s \overline{fs}_{fs} + \sum_i \overline{fe}_{if} + \sum_t (\overline{fm}_{tf} - \overline{fx}_{tf}) = \sum_s \left[\overline{fd}_{fs} \left(\frac{CVA_s}{PF_f} \right) Y_s \right]$$

Market clearance for PFX:

$$\sum_g \overline{x}_g X_g \left(\frac{PFX}{RX_g} \right)^{\eta_g^x} + \overline{fxe} = \sum_g \overline{m}_g M_g + \left(\frac{ROW}{PFX} \right)$$

Market clearance for PNX:

$$\sum_g \overline{xn}_g X_g \left(\frac{PNX}{RX_g} \right)^{\eta_g^x} + \overline{nxe} = \sum_g \overline{mm}_g A_g \left(\frac{CA_g}{PNX} \right)^{\sigma_g^{dn}}$$

B.6 Income Balance

The income balance condition simply indicates that income received must be equated with all expenditures. This condition applies to both demand agents: the representative agent (RA) and rest-of-world (ROW). The income balance condition for both agents then are:

$$RA_i = \sum_g PY_g \overline{y}_{ig} + \sum_{i'} PT_{i'} \overline{br}_{i'i} + PNX \sum_{trd} \overline{br}_{trd,i} + \sum_f PF_f \overline{fe}_{if}$$

$$\begin{aligned} ROW &= PNX(-\sum_{i,trd} \overline{br}_{trd,i}) + \sum_i (PT_i \sum_{trd} \overline{br}_{i,trd}) \\ &+ \sum_f (PF_f \sum_{trd} (\overline{fm}_{trd,f} - \overline{fx}_{trd,f})) + PNX\overline{nxe} + PFX\overline{fxe} \end{aligned}$$