GTAP Data Base Build Automation and V4 Replication Report

Betina Dimaranan
25 January 2000[1]

This document is a completion report on the task of fully automating the data base construction or "build" process that was initiated for the GTAP version 4 data base (v4) and of replicating the v4 data base *after* its public release in October 1998. Full automation and replication of the v4 data base construction process is an essential component in the preparatory work leading to construction of interim data bases and future releases of the GTAP data base (starting with version 5 which is scheduled for public release in August 2000).

A brief description of the data base construction process is provided in the next section. The second section describes the tasks undertaken to fully automate the process. The next section reports on the replication of the version 4 data base using the fully automated construction package. The first fully automated data base construction package is archived and packaged in three zip disks as described in the fourth section. Finally, the last section of the document outlines what is left to be done to further improve the data base construction package.

1. Background on the Data Base Construction Process

Construction of the GTAP global data base is a complex process which includes such activities as cleaning the initial regional input-output (IO) tables; integrating regional IO data with macroeconomic, trade, and protection data; preparing the various parameters used in the model, and the final assembly of the global data base. The various processes involved in data base construction are divided into *modules* which are listed and described in the first two columns of Table 1.

The data base construction procedure is implemented with the use of the *MAKE* program. (The *MAKE* program used in this implementation is from OPUS software, as bundled in Lahey *MAKE* Version 5.2). The linkages within and between modules, i.e. the dependence of a target file on previous intermediate and target files, are specified in *makefiles*. The commands and programs needed to run each module is contained in the *makefile* for that module. The integration of a module to the other modules is dictated by a master *makefile*. The master *makefile* contains

[1] This is a revised version of the original document, now v000doco.ori, dated 10 December 1999. The revision pertains to the recent automation and integration of the laborsplit (*lab*) module into the build process. As of the time of the original document, the *lab* module was not integrated and only its' final output, *labor.har*, was read into the build process.

commands that call on the *makefiles* of the individual modules and commands that prescribe how each module depends on a previous modules' output. The recursive structure is used so that each module can be run independently from its own *makefile* or from the higher level master *makefile*. In the case of sub-modules, the *makefile* for the modules acts as the master *makefile* dictating the interaction between the different sub-modules. Further details on the standards that are being followed and developed in the data base construction process are available from the document "GTAP Data Base: Developers' Reference."

At the time of the final release of the version 4 data base (October 1998), not all of the modules were fully automated and integrated into the build process. In particular, the module on disaggregation of the agricultural sectors in the regional input-output tables (*IODAGG*), the protection module (*PROT*), the CDE parameter calibration module (*PARCAL*), and the laborsplit module (*LAB*) were ran outside the build process and the outputs from these modules were just read into the build process which was specified in two master *makefiles*.

The task of fully automating the data base construction process involves bringing in the *iodagg, prot, parcal,* and *lab* modules and fully integrating them into the build procedure. The replication task involves ensuring that the output produced by each module is the same as the output produced by the said module in the version 4 build process and that the final outputs - the global data base file (*gsddat.har*), the global parameters file (*gsdpar.dat*), and the global sets file (*gsdset.har*) - are the same as the corresponding version 4 final release files. In addition to automation and replication, various minor tasks geared towards the improvement of the build process were also performed. These tasks, limited to those that will have no effect on the data base output, are just part of a longer list of desired revisions to the construction process.

2. Full Automation of the Build Process

The goal in fully automating the build process is to be able to construct the entire data base using a minimal number of master *makefiles*, ideally only one. The achievement of this end requires that all modules be fully automated internally and also fully integrated into the entire construction process such that in running the build job the module will be able to derive the input files that it needs from previous modules and will also produce the output files that subsequent modules require from it.

When the work described in this document was begun, the *iodagg, prot, parcal*, and *lab* modules were not integrated into the build process and were also in different stages of internal automation. In producing the version 4 data base, these modules were ran outside the build process using input files derived from outputs of previous modules and the output files were simply read into the construction process. The automation-related tasks performed on the construction process are described below and also outlined in the third column of Table 1.

2.1 IOINI

The region-specific set information files were put together with each primary region's input-output table in the region-specific subdirectories in this module. Some minor processing of these region-specific set information files used to be in the *iocln0* module.

2.2 IOCLN0

The initial processing to produce of region-specific sets information files, *${reg}_sec.txt*, for use in the *iocln0* and *iodagg* modules was removed from this module. Some other changes were made for replication purposes and this are described in a later section.

2.3 IODAGG

With the introduction of a finer breakdown of the agricultural sectors in the version 4 GTAP data base, the primary regions in the database had to be classified into four groups according to the amount of sectoral detail contained in their input-output tables. The regions were classified as to whether there is: (a) full sectoral detail, (b) full disaggregation of the agricultural sectors, (c) the agricultural sectors are not fully disaggregated but detailed agricultural disaggregation information is available, and (d) the agricultural sectors are not fully disaggregated and only rough agricultural disaggregation information is available. Since the procedure and files required to fully disaggregate the agricultural sector is different for each of the above groupings of regions, sub-modules were created for each classification.

The module, as developed by J. Liu and R. McDougall, is only partially internally automated. It is composed of a group of sub-modules which are integrated together and automated by one *makefile*, as well as three other sub-modules which are not integrated at all with the main group. The main group of sub-modules is composed of:

(a) GENERIC : intended as sub-module for the preparation of executable versions of sub-module-generic TABLO files.
(b) AGRR : for processing of regions whose agricultural sectors are not fully disaggregated and only rough agricultural disaggregation information is available
(c) AGRD : for processing of regions whose agricultural sectors are not fully disaggregated but detailed agricultural disaggregation information is available
(d) 49SEC : for processing of regions with full disaggregation of the agricultural sectors
(e) FINAL : for assembly and final processing of output from *agrr*, *agrd*, and *49sec*.

Table 1. List of Modules in the Build Process, GTAP Version 4 Data Base

V4 Modules	Module Description	Tasks Performed for Automation
SETS	preparation of global sets information	
IOINI	primary-region IO tables and set definition	included region-specific sets information files in the region-specific subdirectories
IOCLN0	cleaning of primary-region IO tables	removed processing of region-specific sets information files
MACRO	preparation of macroeconomic data targets	
IOREP	creation of representative IO table	
IODAGG	disaggregation of IO tables (has sub-modules)	various revisions including the inclusion of *oldtarg* as a sub-module; fully automated and integrated the module
ADDAGG	post-*iodagg* correction	revised and integrated the module
IOCLN1	cleaning of disaggregated IO tables	
IOCOMP	creation of IO tables for composite regions	
IOROB	robustify all IO tables	
TRADE	preparation of international trade data	
PROT	preparation of protection data (has sub-modules and sub-sub-modules)	merged the two sets of sub-modules (by W. Yu and R. McDougall); revised the structure of the module; fully automated and integrated the module
IOFIT	fit IO tables to macroeconomic, trade and protection data	
IOASS	assembly of fitted IO tables	
PARETC	preparation of misc. parameters data	
SUPP	preparation of supply-related data	
PARCAL	consumer demand system parameters calibration	various revisions including the revision of the *gcde.gms* file, TAB and SIF files and the *makefile*; integrated the module
PARASS	assembly of all parameters	
LAB	occupational disaggregation of labor employment data (laborsplits)	revised and integrated the module
DFASS	final data assembly	

The other sub-modules are:

(f) SETS : for the preparation of expanded sets and mapping information for the *iodagg* module
(g) TARG : for the preparation of target production totals which are inputs for the *agrr* sub-module
(h) PRE : for post-*iodagg* revision of the IO structure of the Vietnam IO table

Since the *generic* sub-module was not really operational as a module but served only as a source repository for sets and mapping files, it was eliminated from the module. To fully automate the module internally, the *sets* and *targ* sub-modules were brought into the *iodagg* module. The *pre* sub-module was treated as a separate module called *addagg* below.

The *agrr* sub-module requires estimates of regional production and regional imports of agricultural commodities to serve as targets in the disaggregation process. These data are available in two separate files with regional production estimates sourced from the file output of the *targ* sub-module and regional import estimates obtained from a file local to the original *agrr* sub-module. A sub-module parallel to *targ*, which will produce the file with regional import estimates was created and called *oldtarg*.

To integrate the *iodagg* module with the rest of the build process, the module-generic subdirectories *lcl, in, out,* and *thr*, were introduced. The first three sub-directories have the same function as they have in other modules. The *thr* subdirectory was introduced to serve as a conduit subdirectory for copying output files from one *iodagg* sub-module to another. A *tmp* subdirectory was used for this purpose in the sub-group of the original module.

Various other revisions to improve automation and to conform with data base construction standards were implemented on the *iodagg* module. These include creation of a module-specific macro variable which specifies regional groupings, *mfdisdat,* for inclusion in all *makefiles*. Piping of files from one operation to another was also eliminated so that errors can be detected immediately.

The final *iodagg* module, automated and integrated into the build process, is comprised of the *lcl, in, thr,* and *out* subdirectories and the *sets, targ, oldtarg, agrr, agrd, 49sec,* and *final* sub-modules. The master *makefile* was revised to integrate the *iodagg* module into the build process. Under its present structure, *make* does not have sufficient memory to run the *iodagg* module to completion from the master *makefile*. It is recommended that the module first be run from the master *makefile* so that copying of inputs to the module from the global directory will be done and then run the module internally using the *iodagg makefile* when memory problems kick in.

2.4 ADDAGG

In constructing the version 4 data base, it was found that after the *iodagg* procedure, the

IO table for Vietnam still had a bad cost structure for sector 2. The module developers created a sub-module called "*pre*" to correct the problem for Vietnam and overwrite the disaggregated Vietnam IO table created in *iodagg*.

The *addagg* module was created to bring in the Vietnam IO table correction procedure into the build process. Unlike the *pre* sub-module which deals only with the Vietnam IO table, the *addagg* module reads in all regional IO table outputs of the *iodagg* module and outputs all regional IO tables for the subsequent module even though only Vietnam's IO table is corrected. It is expected that future improvements on the *iodagg* module will eliminate the need for the *addagg* module.

2.5 PROT

The protection module prepares and combines data on various protection instruments to generate region-specific files which contain bilateral protection rates. The protection instruments covered are production subsidies, export instruments (anti-dumping duties, price undertakings, voluntary export restraints, agricultural export subsidies, and MFA rates), and import instruments. The processing of the data is handled in separate sub-modules, roughly by instrument.

The original module developed by W. Yu and R. McDougall was in two major sets of sub-modules which were only partly automated internally within each set and not integrated as a module. The two sets were somewhat parallel to each other. Table 2 reports the sub-modules contained in both sets. The merchandise tariff sub-module (*imtf*) in the first sets was further divided into sub-sub-modules for processing tariff data sets obtained at different dates (some from GTAP version 3 data base). Both sets of sub-modules included a *gnr* (generic) subdirectory which was used as throughput directory for files within the set/module and for input files from and output files to other modules.

The protection data for GTAP version 4 was created from a combination of the sub-modules from both sets. In some cases parallel sub-modules from the two sets were both used -- for example, output from the *aps* and *axs* sub-modules in the second set were used as inputs and further processed in the corresponding sub-modules of the first set (*outs* and *xsub*, respectively). The final set of regional protection files from the *ass* sub-module in the second set used the final output from the *assy* sub-module from the first set but revised them to replace the *mfa* data with those from the *mfa* sub-module from the second set.

Automating the *prot* module involved merging the two sets of sub-modules, automating each sub-module, and integrating the sub-modules internally. Because of insufficient memory to run the entire *imtf* sub-module and its sub-sub-modules under *make*, the sub-module was split into two - the *itx* and the *imt* sub-modules. The *itx* sub-module processes and assembles sets of tariff data into one global tax file and the *imt* sub-module starts from the global tax file and adds agricultural tariff data and information on free trade areas to come up with regional bilateral tariff

files. The *ait* (agricultural import tariffs) sub-module was included as a sub-sub-module under the *imt* sub-module. The last column of Table 2 lists the final set of sub-modules in the internally automated and integrated *prot* module.

Table 2. Protection Module Sub-Modules

PROT Sub-module Description	First Set (Yu)	Second Set (McDougall)	Final PROT Sub-modules
Anti-dumping / Price Undertaking	ADPU		ADP
Output Subsidies	OUTS	APS	APS
Export Subsidies	XSUB	AXS	AXS
Merchandise Tariffs	IMTF	(AIT)	ITX and IMT
Multi-Fiber Arrangement	MFA	MFA	MFA
Voluntary Export Restraints	VER		VER
Assembly	ASSY	ASS	ASS

Since the *itx* and *imt* sub-modules are composed of sub-sub-modules, to facilitate internal automation within these sub-modules, *lcl, in, thr*, and *out* subdirectories were included in these modules. The role of the *lcl*, *in*, and *out*, are parallel to their function in sub-modules and the *thr* subdirectory serves as throughput for copying output files from one sub-sub-module to another. Under the *prot* module proper, similar to what was done in the *iodagg* module, module-generic subdirectories *lcl, in, out,* and *thr*, were also introduced in order to integrate the module with the rest of the build process. The master *makefile* was revised to integrate the *prot* module into the build process.

Aside from automating and integrating the module internally and integrating it with the rest of the build process, various other revisions were performed on the module. These include: changing the format of input files to the ADP sub-module; revision of *makefiles* to allow the use of macros and to conform to other data base standards; and revision of *makefiles* to avoid spurious dependencies and excessive remaking.

Under its present structure, *make* does not have sufficient memory to run this big *prot* module to completion from the master *makefile*. It is recommended that the module first be run from the master *makefile* so that copying of inputs to the module from the global directory will be done and then run the module internally using the *prot makefile* to make some internal target. Another run from the *makefile* followed by an internal run may have to be done before the module can produce its final output files.

2.6. PARCAL

The CDE parameter calibration module, developed by J. Liu and R. McDougall, received for integration to the build process is an improved version of the PARCAL module which was used to generate the calibrated parameters for the version 4 data base. Although fully automated internally, some internal revision was done to the module before integrating it with the rest of the build process. These revisions include: elimination of redundancies in the sets files; renaming of output files; modification of the *GAMS* calibration program, *gcde.gms,* to change the variable that it will output (*subpar* instead of *alpha*); and elimination of unnecessary steps in the process. To accommodate the revisions, *TABLO* and *SIF* files were revised, renamed or eliminated.

2.7. LAB

The *laborsplit* module, developed by J.Liu, produces the shares or splits of skilled labor and unskilled labor in the total labor payments for each sector and region in the version 4 data base. This data is used in the final module (*dfass*) to split the labor factor into two categories - skilled and unskilled - by applying the shares to the payments to the labor endowment for each sector j and region r, EVFA("labor", j, r). There is considerable preprocessing work involved in generating the data used as inputs to the module but this was not integrated into the module. Much of it is based on the work documented in "Disaggregating Labor Payments by Skill Level in GTAP," GTAP Technical Paper Number 11, authored by Jing Liu, Nico van Leeuwen, Tri Thanh Vo, Rod Tyers, and Thomas W. Hertel. The final data inputs integrated into the *lab* module are: (1) the predicted skilled labor payment shares for 45 regions and 30 sectors (version 3 sectors with agriculture aggregated) which resulted from some estimation procedure, and (2) actual skilled labor payments shares for 12 regions and 30 sectors. Although fully automated internally, some minor revisions were done to the *makefile* and the final output file was renamed *labor.har*.

2.8. Master Makefiles

Two master *makefiles* were used to create the version 4 data base – *1.mk* covered the pre-fit and fit (*IOFIT*) modules and *0.mk* covered the post-fit modules. Ideally the goal is to be able to fully automate the process so that the entire data base can be created using one master *makefile*. However, with the complete automation and integration of the *iodagg*, *prot, parcal,* and *lab* modules into the build process, it became necessary to split the construction process into four master *makefiles* to avoid *make* memory problems. For example, cases were observed where *make* runs out of memory while still in the stage where it checks that files in the dependency tree are up-to-date. The final master *makefiles* and the modules that they cover are as follows:

(a) 3.mk : sets, ioini, iocln0, macro, iorep, iodagg, addagg, iocln1, iocomp
(b) 2.mk : iorob, trade
(c) 1.mk : prot, iofit
(d) 0.mk : ioass, paretc, supp, parcal, parass, lab, dfass

Testing the complete automation of the data base construction process is done by emptying the contents of all *in*, *thr*, *wrk*, *out*, and *dmp* directories and using the master *makefiles* in the above sequence to create the database. The archiving sections in the original master *makefiles* were eliminated and replaced instead with one master *makefile*, *acv.mk*, for archiving the entire sets of data base construction files. Although the GTAP construction process is fully automated in the sense that the entire construction process is integrated into four *makefiles*, memory problems will still be encountered when running the *iodagg* and *prot* modules. As indicated in the sections pertaining to these modules above, it will be necessary to run the modules internally after the copying of the input files from previous modules has been done by the master *makefile*.

2.9. Other Modifications

As mentioned in various sections above, modifications were performed on the modules in order to improve automation and to adhere to some data base construction standards. The more substantial among these modifications were: (1) the elimination of command piping, especially in the *iodagg* module, so that errors are caught immediately; and (2) the revision of *makefiles* to reduce spurious dependencies and excessive remaking, especially in the *prot* module.

Another significant modification in the build process is the addition of local, *lcl*, sub-directories in each module and\or sub-module. The *lcl* sub-directory serves a repository of input files that are used solely in that module/sub-module. In running the module, the master *makefile* directs the copying of the input files from the *lcl* subdirectory into the *in* subdirectory. Having a *lcl* subdirectory in each module is useful in identifying the input files needed by the module which it will not obtain through the build process from an earlier module. It is also handy during the testing stage in that it allows for the deletion of all files from the *in* subdirectory before running the module.

The modifications done in this round of work are only a subset of the modifications required for a more efficient automated data base construction process, as outlined in the final section of this document. In particular, the modifications do not include those that might affect the output files produced by the module such as to compromise the replication of the V4 database (as discussed in the next section). In addition, the documentation (readme files, etc.) for each module, where available from the original module developers, have not been revised to reflect the changes that have been made.

3. Replication of the GTAP Version 4 Data Base

After fully automating and integrating the data base construction process, it was found that the resulting final data base was far from being the same as the final version 4 data base. This necessitated a careful and tedious investigation of the entire process with an end to replicating at key points what was actually produced in the Version 4 build process. The results of this replication is reported in this section.

All the modules for which numerical differences between the old GTAP version 4 results and the new results from the automated data base construction package may arise were examined. This mostly involved those modules which deal with the regional input-output tables. *GEMPACK's CMPHAR* program was used to compare the header array files from the new results against those from the old results. Among other indicators, *CMPHAR* reports the magnitude and the position of the largest absolute difference between two headers from the old and the new file. In the archived files, the subdirectories *oldin, oldwrk,* and *oldout,* if present, contain the old files and the files used to compare the old results with the results (*makefile* and master *sti* files).

Using the largest absolute differences reported by *CMPHAR,* the headers in each file from the old and the new results were compared to check that the differences are not economically significant. A more stringent criterion established for determining that replication of the results has been achieved is that the two numbers which *CMPHAR* reports as having the largest absolute difference for each matching header from the old and the new results, should agree to at least five significant figures. Using this criterion, the old and new output files in each module are compared and if the criterion is not met the sources of the differences are traced by comparing the input files and, if needed, the intermediate files in the *wrk* directory.

The results of the replication process for each relevant module are summarized in Table 3. In particular, it reports what had to be done to replicate the results obtained in the Version 4 build process using the five-significant-figure criterion mentioned above. Since the modules were developed independently from each other during the actual V4 build process, module developers had to use whatever latest version of the inputs files, especially input output tables from the previous module, were available. It seems that there were cases when the input IO tables were not updated in the final run of the construction build process. Hence, after completely automating of the data base build process, it was necessary to break the link between a module producing processed IO tables as output and the subsequent module where the IO tables are needed as inputs. For cases where the link between two modules had to be broken, the output files from one module were not used as inputs by the module that requires them but instead old input files used in the original V4 module were used. Where necessary, the old input files from the V4 process were brought into the *lcl* subdirectory of the module and the master *makefiles* were temporarily altered to direct the module to use these older files instead of picking up the input files from the build process.

Table 3. Report on Replication, by Module, GTAP Version 4 Data Base

V4 Modules	Remarks on Replication
IOCLN0	* replication achieved only after some internal revision of the module so that the *taxrev* and not the *negrev* program was used in the process - the original *makefile* and master *sti* files indicate the use of *negrev* (*nrv*) files but the intermediate *sti* files indicate the use of *taxrev* (*trv*) files * creation of summary report file suspended due to problems with *DJGPP bash*
IODAGG	* used old input files, *$(reg)_cln.har*, as used by J. Liu, instead of output from *iocln0* * broke the link between *iocln0* and *iodagg*. * replicated all sub-modules except the *final* sub-module - could not check further since old intermediate (*wrk*) files were not retained by module developers.
ADDAGG	* used old input files, *$(reg)_dag.har*, instead of output from *iodagg* * broke the link between *iodagg* and *addagg*
IOCOMP	* creation of summary report file suspended due to problems with *DJGPP bash*
TRADE	* the module calculates and uses shares from the IO tables *$(reg)_rbt.har* which are output of *iorob* module. Although output files from *iorob* were replicated, the shares calculated from them were not so the *trade* module was revised to use the old *wrk\iodatshr.har*. The master *makefile* and trade *makefile* were temporarily altered to read in *iodatshr.har* as a local file.
PROT	* used old input files, *$(reg)_rbt.har*, as used by W. Yu in the *prot\imt\cflate1* sub-module, instead of output from *iorob* * used old input files, *$(reg)_rbt.har*, as used by R. McDougall in the *prot\ass* sub-module, instead of output from *iorob* * broke the link between *prot* and *iorob*
IOFIT	* although the *$(reg)_rbt.har*, *$(reg)_trd.har* and *$(reg)_ptt.har* files which are outputs of the *iorob*, *trade* and *iodagg* modules, respectively, were satisfactorily replicated to 5 significant figures, to minimize the sources of difference in this simulation module, the old V4 versions of these input files were used * broke the link between *iofit* and the *iorob*, *trade*, and *prot* modules * the differences in the resulting output files that occur are small and not economically significant
IOASS	* used old *iofit* output files, *$(reg)_tb.har*, as inputs * broke the link between *ioass* and *iofit*
PARASS	* used the old *parcal* module output file, *gcde.har*, as input * broke the link between the *parass* and *parcal* modules
DFASS	* used old *prot* module output files, *$(reg)_ptt.har*, as inputs, although this may not have been really necessary * broke the link between *dfass* and *prot* – may not have been necessary

4. Archived Automated GTAP Data Base Construction Package (Revision 0.0.0)

The files associated with the fully automated GTAP data base construction process are archived in three zip files in three zip disks. The three zip files each contain zip files for each module, as listed in Table 4 below. There is also a readme file which contains brief instructions on how to unzip all files, a *makefile, unacv,mk,* which should be used for unzipping all the files from a desired root directory, and a copy of this document.

Table 4. Archiving Information for GCP v.0.0.0

GCP v.0.0.0 disk 1	GCP v.0.0.0 disk 2	GCP v.0.0.0 disk 3
v000_1.zip : sets.zip ioini.zip iocln0.zip macro.zip iorep.zip iodagg.zip addagg.zip iocln1.zip iocomp.zip iorob.zip	v000_2. zip : trade.zip prot.zip iofit.zip ioass.zip paretc.zip supp.zip parcal.zip parass.zip	v000_3.zip : lab.zip dfass.zip glob.zip mfbefore 0.mk 1.mk 2.mk 3.mk acv.mk readme unacv.mk v000doco.ori v000doco.wpd (this document)

5. What's Next

Since the principal objective for this round of work was to automate and integrate the *iodagg*, *prot*, *parcal* and *lab* modules and then to replicate the V4 data base results, various desired modifications for the improvement of the automation process were not addressed. These modifications will be addressed in subsequent revisions to the data base construction process. After implementing a version management system, the planned data base revisions include the following:

(a) removal of replication fudges, resetting of the links between modules
(b) completion of the automation process including improvements in structure
(c) implement regional flexibility
(d) integrate new or revised modules (e.g. energy, services trade)
(e) bring in new or revised regional IO tables into the data base

A shift to a different *MAKE* implementation (to a newer version of *OPUS make* or to GNU *make*) is desired in order to reduce the *make* memory problems encountered in the automation process. The document, "GTAP Data Base: Developers' Reference," will be maintained and the data base construction standards discussed therein will be updated and adhered to in the data base construction process.