B Global Data Assembly

Thiago Simonato

This chapter of the documentation has three major sections. The first section provides a brief summary of the entire GTAP Data Base construction process. Section B.2 describes the actual procedures involved the last stage of the process, the final data assembly module. Section B.3 summarizes the standards adhered to in the data base construction process.

B.1 Data Base Construction Summary

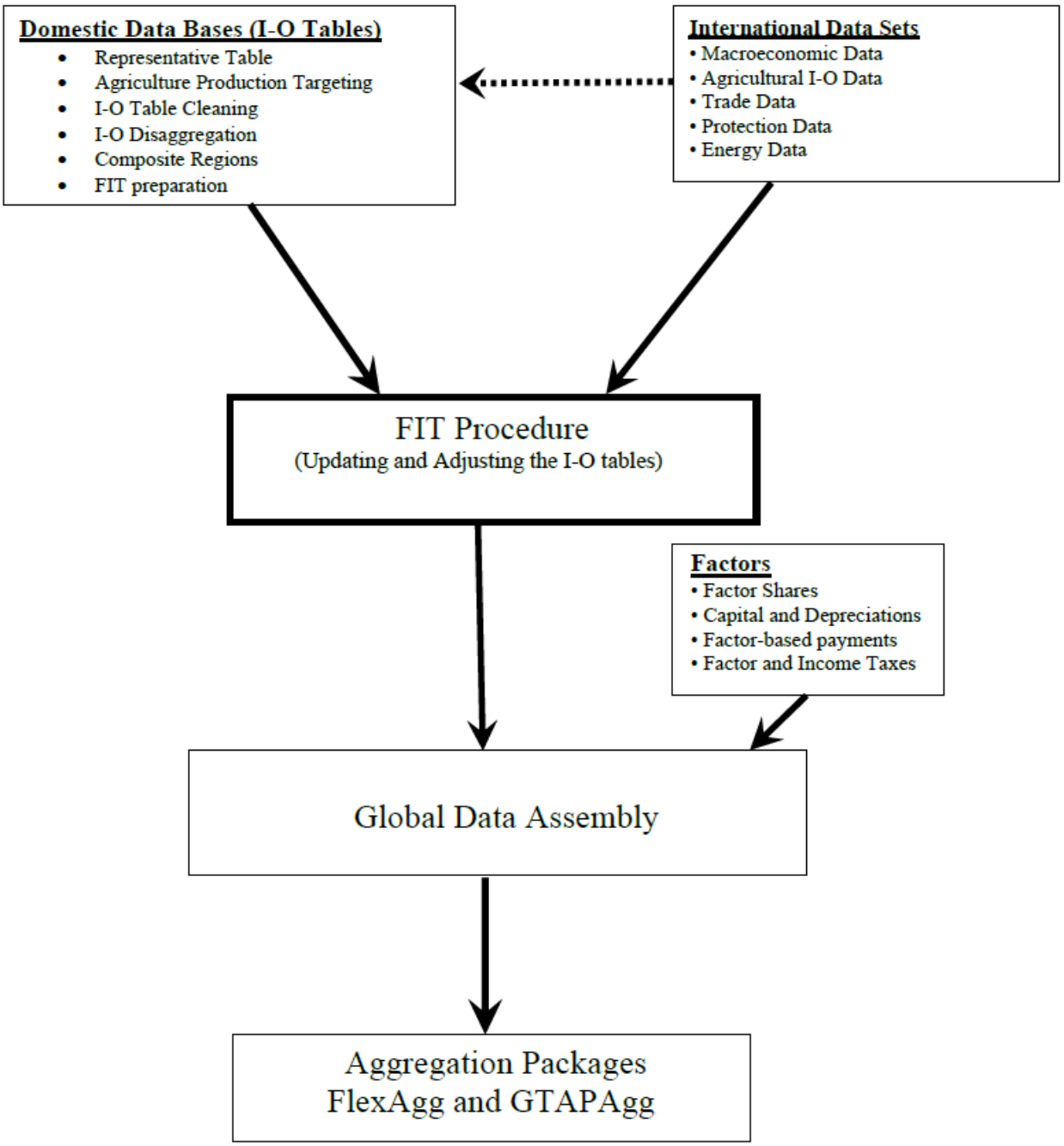
The data base construction procedure is devoted largely to the construction of the main global data base file. The procedure also generates a number of files such as: global sets, parameters, energy volumes, data summary (BaseView), and tax rates summary. The whole process is divided into sub-processes or modules with each module designed to handle a well-defined component of the data construction procedure, e.g., I-O disaggregation, trade data, etc. (see more about modules in Section B.3).

The construction of the main data base file involves the preparation of global sets and mapping files used in the construction process; the preparation of domestic databases and international data sets; updating the regional data bases to match the macroeconomic, trade and protection data for each reference year; and final data assembly. Figure 17.1 provides a simplified illustration of the data base construction process.

The domestic data bases or input-output tables undergo some preliminary evaluation upon receipt from data contributors. A representative I-O table is constructed from the I-O tables which have full GTAP sectoral disaggregation (Chapter 9.B). The contributed input-output tables which have satisfied the guidelines for contributors then go through checking and cleaning procedures (see Chapter 8). The I-O tables which do not have full agricultural and/or non-agricultural disaggregation then go through the I-O disaggregation procedure (Chapter 9.B). Input-output tables are constructed for the composite regions, i.e., GTAP regions for which there are no contributed I-O tables (Chapter 9.D). The domestic data bases are then prepared for updating and adjustment procedure, i.e. the FIT process (Chapter 17.A). Some international data sets are required in the processes pertaining to the I-O tables (thus the arrow from the international data sets box to the domestic data bases box in Figure 17.1). These include the macroeconomic data (GDP and GDP per capita) and the agricultural I-O data used in the disaggregation process.

The international data sets obtained from external data contributors are also processed. The procedures involved are laid out in the previous chapters. The international data sets are the: macroeconomic data (Chapter 7), agricultural I-O data (Chapter 9.A and B), trade data (Chapter 10), protection data (Chapter 11), and energy data (Chapter 12). Raw or semi-processed data at disaggregate country levels are obtained from data contributors. Further processing, including extending the data to all standard countries, filling in missing values, and aggregation to the GTAP regional and sectoral classifications, is performed.

Figure 17.1: GTAP Global Data Base Construction Procedure



Since the reference periods of the I-O tables vary, they are updated and reconciled to a common base year(s) of the GTAP Data Base (i.e., 2004, 2007, 2011, 2014, 2017, 2019 and 2023 for the GTAP 12 Data Base) using the macroeconomic data set and the other international data sets such as trade, protection, and energy using the FIT procedure (Chapter 17.A). The regional data bases are then assembled to construct an interim global data file.

Prior to the final assembly module, factor shares data for use in adjusting the payments to land, labor, and capital in agriculture are also prepared. For primary factor shares in the natural resource based sectors, a proportion of the earnings of labor and capital is reallocated to natural resources to achieve target supply elasticities (Chapter 16). For each region and sector, labor payments are disaggregated using labor earnings shares from the labor data set (Chapter 14.B).

B.2 Final Assembly Module

The final data assembly module is where the various international data sets and domestic data bases are put together to form the main global data base. It is also where the global sets, parameters, energy volumes, and tax rates files undergo some minor, final processing. The assembly procedure for the main global data base involves:

— disaggregation of the labor payment data using payment shares generated in the estimation procedure documented in Chapter 9.B;

— revision of factor employment data for primary agriculture and natural resource-based sectors using primary factor shares documented in Chapter 14;

— merging the interim global data file (from FIT procedure, see Chapter 17.A) with the bilateral trade data (Chapter 10), protection data (Chapter 11), and capital stock and depreciation data (Chapter 7)

— re-balancing the data base to correct imbalances between domestic product supply and usage and between income and disposition—further checks on data base for sign violation, imbalances, zero-divide problems.

The central task in the data assembly procedure is merging the interim global data file with the trade, protection and capital stock data. Various data adjustments are done, these include:

— international trade and transport margins data and capital stock and depreciation estimates are incorporated into the data base

— the implied revenues/payments associated with the trade-related protection measures are calculated

— the value of trade at basic prices are calculated from the protection revenue data (for imports and exports) and trade data at CIF prices

— the factor payments data are adjusted to incorporate land- and capital-based payments

— the value of firms' purchases of intermediate inputs are adjusted, as necessary, to eliminate cases of very large subsidies on extremely small base value flows

— regional savings is calculated and included as a data array

— inclusion of labor-based agricultural support payments for non-EU OECD member countries (for more details, see Chapter 11)

The global sets, parameters, and energy volumes data file, which are generated in previous modules undergo some minor processing in the final assembly module. The global data summary or BaseView file and tax rates summary file (see Chapters 4 and 5) are also generated in the final data assembly module.

B.3 Data Base Construction Standards

Since GTAP 6 Data Base, the data construction procedure has been automated, thereby making it replicable and flexible. This section provides a brief discussion of the standards that have been introduced in the data base construction process to better handle the large and complex task of creating the GTAP Data Base.

B.4 Flexibility

There are two main aspects to the flexibility that we want to achieve in the data base construction process. Firstly, we want to make it easy to revise the data base when we get new source data, and secondly, we want to be able to easily revise the regional classification. To make it easy to revise the database when we get new source data, we keep data and programs in separate files. We count data as not only economic flow data but also the sets and mapping information. To make it easy to revise the regional classification, we rely partly on the practice described above, but also on another practice, that of encouraging contributors of multi-country data sets to provide the data on a country basis rather than on a GTAP region basis. This allows us to revise the GTAP regional classification without going back to data contributors for new reclassified source data.

Regional flexibility is achieved when new regions can be added by simply including the new region(s) in a list or regions used in the data construction process and supplying the additional country's I-O table. Collecting international data bases at the country level helps achieve this objective. The international data sets are then mapped or transformed to a standard set of countries. A mapping between the standard set of countries and the set of GTAP regions is then used to aggregate the data. The mapping file is revised when a new region is introduced in the GTAP data base. Regional flexibility is also facilitated by ensuring that there is no hard-coding of regions in the program codes. The set of region and mappings are read from files that are generic to the construction process.

B.5 Build Management

We automate the construction process using the program *make*. The *make* program identifies the commands needed to create or update the data base, runs those commands, and displays an error

message and stops if it encounters an error in running them.

To tell *make* how to create the data base, we maintain a *Makefile* showing which target and intermediate files depend on which intermediate and original files, and the commands needed to create the target files from the files on which they depend. The *Makefile* also contains brief comments, providing basic documentation of the build procedure.

To keep the *Makefile* manageable and to support a modular structure, we use *make* recursively. The master *Makefile* does not contain any of the commands directly used to create the data base; instead it shows how the different modules depend on each other—how each module outputs are another module's inputs—and contains commands to run make on each module. These recursive make commands read information from each module-specific *Makefile*.

In practice, we do not bring the entire construction process under build management. More specifically, we do not bring under build management some procedures involved in the initial conversion or formatting of original data files. We do, however, automate these steps as far as practicable. We confine these unmanaged procedures to initial file conversion and formatting. Their function is to convert the data to GEMPACK-readable form, or some other form which we can handle automatically, as directly as possible. All substantive data processing, as opposed to formatting, is done under build management.

B.6 Reproducible Pipeline Orchestration

In recent versions of the build procedure, the construction process is organized as a staged pipeline with explicit hand-off contracts between stages. Concretely, the top-level build driver treats the overall workflow as a directed acyclic graph (DAG) of targets, where each stage materializes its outputs into well-defined staging directories before the downstream stage is allowed to run. This architecture promotes reproducibility by turning historically implicit data dependencies ("files that happen to exist in some folder") into explicit build-time prerequisites.

A key design decision is that agricultural production targeting (APT) is treated as a first-class stage in the construction pipeline, rather than as an ad-hoc adjustment outside the build (see Chapter 9.A). Operationally, the pipeline inserts APT between two make-managed runs of the construction procedure:

- a *partial run* that prepares the macro and I–O structure and then fits region-wise I–O tables (with internal consistency/target checks), delivering a clean, fully disaggregated (when required) set of tables in a stable format suitable for targeting; and
- a *final run* that consumes the APT-adjusted tables as canonical inputs, replays the full construction sequence, and assembles the final database deliverables (sets, parameters, views, and validation reports).

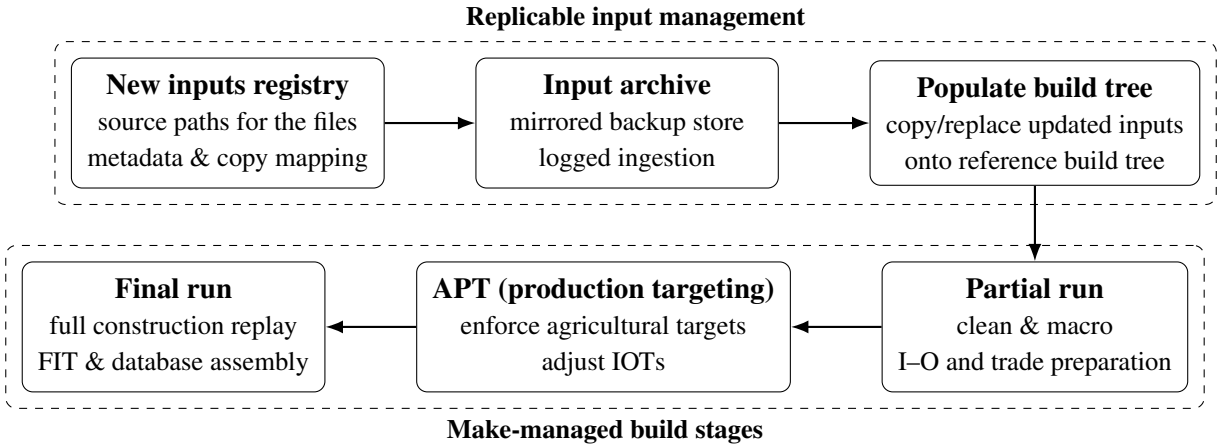
These two runs map onto selected blocks of the construction process summarized in Figure 17.1, with a deliberate restart around APT. The *partial run* executes the upstream preparation

steps required to obtain fit-ready regional tables (including cleaning/disaggregation as needed and the preparation of macro/trade/protection/energy inputs up to region-wise I–O fitting). The *final run* then restarts the construction workflow using the APT-adjusted regional tables as the authoritative inputs: it re-executes upstream steps as needed to maintain internal consistency under the new targets, and additionally completes the downstream procedures shown in Figure 17.1, including the FIT, reconciliation and the final global data assembly outputs.

This design is motivated by the requirement that all regions be subjected to a consistent set of updated targets. In practice, the targeting routine is designed to apply these targets as broadly as possible: it can adjust selected parameters to reconcile the targets with the underlying accounts and to obtain the best attainable match. Targets are therefore only relaxed or left unapplied in genuinely exceptional situations, such as infeasibilities implied by inconsistent source data or binding accounting constraints.

Figure 17.2 summarizes this orchestration at a high level. The emphasis is on an automated, replicable workflow in which validated inputs are curated and archived, and then used to update a reference build tree via a controlled copy/replace step before the build is executed through a small number of stable stages. The diagram is intentionally generalized: it captures stage boundaries and data hand-offs, rather than enumerating every file-level prerequisite.

Figure 17.2: Conceptual build workflow for input management



To support incremental rebuilds without sacrificing correctness, the pipeline uses stamp or flag artifacts to represent completion of coarse-grained phases. These stamps act as synchronization points and reduce unnecessary re-execution while keeping dependency boundaries explicit.

B.7 Input Provenance and Automated Ingestion

External inputs are treated as first-class, versioned dependencies. Instead of manually copying collaborator-provided files into the build tree, the procedure maintains a machine-readable input registry (a CSV log) that serves as an *active source* for the build: it enumerates the authoritative source paths of new-build input files, along with the corresponding destination locations in the

build tree and associated metadata (e.g., reference year, last update date, and validation status). The build driver reads this registry programmatically to locate the current inputs and perform the required copy/replace operations in an auditable and repeatable manner.

The build workflow includes automation for (i) pulling the registry, (ii) filtering inputs by year and validation status, (iii) populating a local backup store that mirrors the build-relative paths, and (iv) restoring the backed-up inputs into the build tree in an automated and auditable way (with explicit exclusions for directories marked as non-restorable). This establishes a basic provenance layer: input acquisition is repeatable, the applied inputs are traceable, and transfer operations are logged for later inspection.

B.8 Parallel Execution Model

Parallelism is exploited at multiple layers. At the build-orchestration level, independent targets may be scheduled concurrently by *GNU make* when invoked with parallel job execution (e.g., via the `-j` option), subject to the dependency constraints.

At the module level, region-wise computations are parallelized using bounded concurrency. Region processing is decomposed into per-region command sequences materialized as separate batch jobs, executed with a fixed upper bound on the number of active workers. The implementation uses explicit concurrency control to avoid races over shared mutable state (e.g., pre-compiling common executables once, generating shared header files once, and ensuring that per-region intermediate filenames are disjoint). Execution is instrumented with per-region logs, a global scheduler log, retry logic for transient failures, and timeouts to prevent deadlock-like "stuck worker" situations.

B.9 Modular Structure

We implement the data construction process as a collection of modules arranged in a directory tree. For example, there is a trade module where merchandise and services trade data is prepared (Chapter 10). There is a module wherein the I-O tables for the composite regions are created (Chapter 8.D). There is also a FIT module wherein the I-O table for each region is reconciled with the trade, protection, and energy data (Chapter A). Each module may be run separately or collectively, under the master *Makefile* program.

The data base construction root directory contains the master make description files, various top-level module directories, and a module-generic directory. The root directory contains no data or program files (other than the make description file); these are all held in subdirectories. Modules may contain sub-modules; those that do are laid out similarly to the construction root directory, with all data and program files relegated to module-specific subdirectories and a module-generic subdirectory.

We create a standard module directory structure Within each bottom-level module. This

structure collects files according to their function and treatment within the module. The module root directory contains as many as are needed of the following basic files and directories (directories are marked by an appended slash character /):

— Makefile : make description file

— lcl/ : data input files—files which are local or specific to the module

— in/ : data input files—include generic sets and mapping information files

— src/ : program source—includes TABLO source code, GAMS code and command files

— wrk/ : working files—files that only have an intermediate role within the module

— dmp/ : dumpable reports—includes log files

— out/ : data output—final output files or files required for use in other modules

B.10 Version Control

Version control systems have been implemented in the data construction process. A publicly available software, Concurrent Versions Systems (CVS), is used with the program files. Under this system, each program file is stored in a repository. Modifications to each file, corresponding to a version of the file, are also stored. Each version is tagged and can be retrieved. The use of CVS enables more efficient use of computer disk space since only the latest version of each program file is actually stored as opposed to completely storing all versions of a file.

The Data Version System (DVS), developed by Robert McDougall, is used for retrieving input data files (including header array files) since CVS does not handle HAR files very well.

The version control systems are very important for quality control in the complex data base construction procedure. They enable the developer to start from a clean build each time, i.e. to start with an empty directory structure and populate the directory, in automated fashion, with the latest program code files and also with the latest version of the input files. The data developer can also readily recreate a previous version of an output data file or rebuild an earlier version of the data base by selecting from the version archive system the appropriate set of files used to create that version.

B.11 Common Tool Set

The data base construction process uses a common tool set to spare developers the need to search for and re-implement tools already on hand, learn multiple versions of the same tool, or maintain multiple version of the same tool; and to make it easy to use the same tool set as originally used when replicating old builds.

For data processing, the tool of choice is the GEMPACK software suite, which is also used to implement the standard GTAP model. This is because most of the existing programs in the data base construction package use GEMPACK.

For build management we use *make*. For formatting and text processing, we use *sed*, *awk*, and *perl* since these are all available as free software in implementations of excellent quality. The programs *awk*, and *perl* overlap in function.

For batch jobs, where these are not conveniently handled with DOS batch files, we use *bash* and *shell scripts* written for *bash*. However, there should be little need for this since for the most part, we can run the necessary command sequences direct from the make file.

References

Corong, Erwin et al. (June 2017). "The Standard GTAP Model, Version 7". In: *Journal of Global Economic Analysis* 2.1, pp. 1–119. DOI: 10.21642/JGEA.020101AF. URL: https://jgea.org/resources/jgea/ojs/index.php/jgea/article/view/47.

James, M. and R. A. McDougall (1993). *FIT: An Input-Output Data Update Facility for SALTER*. SALTER Working Paper 17. Canberra: Australian Industry Commission.