Running GIMF – from Calibration to Simulation

Only one file needs to be run – *driver.inp*. *Driver.inp* will run all the other necessary files.

Summary

Run *driver.inp* –

1) It runs *gimf1.src* to create the model
2) It runs *startvals.src* (macro) to create the database.
3) Simulate the steady-state database.
4) *dac.src* is used to calibrate the steady state meaningfully in one time period.
5) Also uses *flipmod.src* (and *changesym* commands) to calibrate parameters using other information (i.e. GDP shares).
6) Create a long steady-state database with *exogvals.src* and re-simulate the steady-state model.
7) Calibrate the dynamic model parameters (nominal rigidities and real adjustment costs) meaningfully.
8) Test the model using *modeval.src*.
9) Using *simulate roots* and *lkroots* check the eigenvalues to understand how the model simulates.
10) *runshocks.src* generates tables of the calibration and does shocks to the model (either impulse responses or forecast scenarios).
    - It runs the shock in steps (i.e. a 2% shock may be run as four 0.5% shocks, one on top of the other) or by using the linearised version of the dynamic model.
    - It runs *graphs.inp* in FAME, which builds tables from the FAME procedure file *graphs.pro*.

The Details

The driver file for the largest model in use (either the four-country NBER model, or any other five country version) has the following sections.

1. Create Starting Values Database "start" and Build the Model Code

This uses *gimf1.src*, a TROLL macro.
Before building the model, the FAME database *start.db* is erased and then created. The list of countries and their names are then defined, as well as the numeraire country (the "nth country"), which used as a point of comparison for bilateral exchange rates, and the currency for the one internationally traded bond.

Next, you must specify the leads and lags you want to you when constructing a variety of tax bases and other variables, all of which are moving average processes (either centred or not; your choice). The definition of each variable is documented right in the driver file.

You can then build the GIMF model, from the TROLL macro, *gimf1.src*.

"compile gimf1.src" builds the TROLL-readable .PRG file (mod1.prg). "&gimf1" runs the program file, as long as you follow it with at one item for the sector list of six items (in any order). This are the sectors and features you wish to include in the model besides the core Blanchard-Yaari overlapping generations framework and the tradable production sector:

1. NONTRADE - Nontradables sector.
2. OIL - Oil sector.
3. UNION - Labour unionization (defines the producer-paid wage).
4. IMPORTS - Import Agents Sector (defines sticky real imports).
5. RETAIL - Retailing sector.
6. ACCEL - Bernanke-Gertler-Gilchrist financial accelerator sector.

An example model, including all items except nontradable, would be created this way:

&gimf1 oil imports retail union accel;

Then you create an input file for each model (steady-state and dynamic) – *ssgimf1.inp* and *gimf1.inp*.

Finally, the main data file, *startvals.src*, is inputted into TROLL. It is the TROLL macro file that loops over each country (every piece of data is listed separately, as a DO command).

2. Create the Initial Steady-State Simulation

Copies *start.db* to *startss.db* and simulates *ssgimf1.mod* with that data from 1910a to 3000a to create a steady state over all time, that has not yet been calibrated to a specific calibration. This calibration is merely a symmetric calibration we know will work and allow the steady-state model to solve.

3. Recalibrate to Create the Steady-State Base Case

This section calibrates all the variables to exact values, but by using mainly *dac.src*, a TROLL macro. The methodology is referred to as the "divide-and-conquer (DAC) algorithm" – instead of moving from value A to the new value Z, which TROLL may not be able to solve for, the calibration is moved in steps from A to Z via B, C, etc. so that the model always has a viable and stable solution (provided that the new value of the parameter does not break some restriction – i.e. being greater than one, when it should be bounded between zero and one). The calibration is done in one time period only (usually 1970a) so that even large models will simulate rapidly.

A typical *dac* call is:

&dac 1910a 7 ss 2.5 xi_c_CA, 2.5 xi_c_US;

Here we are calibrating the two countries' elasticity of substitution between domestically produced tradables and imported tradables.
&dac calls the macro
1910a is the single period in which the steady-state model is recalibrated
7 is the number of pieces the recalibration takes place. For example, assume the initial value for all variables is 1.8. Then the model will be run consecutively with calibrations of the parameter at 1.9, 2.0, 2.1, 2.2, 2.3, 2.4 and 2.5.
ss is the alias of the database from which the model draws and writes its simulation data
2.5 xi_c_CA is the value and the name of the parameter to be calibrated. Other parameters can be calibrated concurrently by stating them in the same fashion on the same line, separated by a comma.

This section calibrates the most general parameters followed by more specific ones, and ending with the elasticities.
This means:

1) The world real interest rate;
2) Very general parameters such as the depreciation rate, the inflation target, and related to the Blanchard-Yaari framework;
3) Some bias parameters;
4) Government debt;
5) GDP shares related to government;
6) The frequency of the model (annual or quarterly);
7) The oil sector;
8) Bias parameters relying on the world trade matrix;

9) Elasticities of substitution for demand and production

When calibrating parameters that include the word _annual, you can always calibrate the parameters to annualized value, even in a quarterly model. So depkbar_annual, the depreciation rate, will always be:

&dac 1910a 2 ss 0.10 depkbar_annual_CA, 0.10 depkbar_annual_US;

for a 10% annual depreciation rate (or 2.5% quarterly depreciation rate in a quarterly model).

We do not use the simulation version of *ssgimf1.mod* to calibrate. Rather, we are constantly shifting towards a calibration version, that changes throughout the process, called *ssgimf1flip.mod*, since we flip the specification of variables. That is, we exogenize formerly endogenous ratios, while we concurrently endogenize formerly exogenous parameters.

We can do this using the TROLL command *changesym* to change the symbol designations to exogenous (*'x*) and endogenous (*'n*):
changesym 'x ZZ_RR_US
changesym 'n bbetabar_US

where we are exogenizing *ZZ_RR_US*, the real U.S. interest rate, and endogenizing *bbetabar_US*, the rate of time preference in the United States.

For the trade matrix we use the TROLL macro *flipmod.src*. This allows us to flip the model – model parameters become endogenous, and paired usually with ratios to GDP for which we have a long-run view. This normally means bias parameters, for which we have no real information are flipped with ratios to GDP – this is always true for trade shares, and sometimes for investment to GDP (with the bias towards capital in the production function).

An example is:

&flipmod SSGIMF1flip
IMPORTS_T_RAT_CA alphathbar_CA,
IMPORTS_C_RAT_CA alphachbar_CA,
IMPORTS_I_RAT_CA alphaihbar_CA,
IMPORTS_T_RAT_US alphathbar_US,
IMPORTS_C_RAT_US alphachbar_US,
IMPORTS_I_RAT_US alphaihbar_US;
filemod SSGIMF1flip;
usemod SSGIMF1flip;
simulate;
&dac 1910a 10 ss
13 IMPORTS_T_RAT_CA, 18 IMPORTS_C_RAT_CA, 6 IMPORTS_I_RAT_CA,

0.81 IMPORTS_T_RAT_US, 1.6 IMPORTS_C_RAT_US, 0.18 IMPORTS_I_RAT_US, 6.55 POPSIZE_CA, 93.45 POPSIZE_US;

usemod SSGIMF1flip; simulate;

&flipmod endogenizes the import ratios (i.e. *IMPORTS_T_RAT_CA*) and exogenizes the bias towards tradable goods in the associated CES aggregator (i.e. *alphathbar_CA*)

Overall, this section will create and use consecutively the FAME databases *ss1.db* and *ss2.db*.

5. Creating the Full Steady-State Simulation

This section extends the steady state to run from 1910a to 3000a, instead of only being one point in time (1910a). The TROLL macro *&exogvals* extends the exogenous data and parameters to last over the full time horizon, and then by doing a TROLL simulation, the endogenous variables are extended as well. The end result is put in the FAME database *ss3.db*.

6. Removing the Unit Roots

All the unit root variables are removed, so that the dynamic model is ready for testing of its eigenvalues, and any extraneous clutter is removed.

7. Set the Dynamic Model Parameters That Do Not Affect the Steady State

Using DO statements in TROLL, those parameters which only appear in the dynamic model of the GIMF are assigned their proper values. This is done in the FAME database *ss4.db* (a copy of *ss3.db*).

8. Checks of the Model

ss4.db is copied to *ss.db*. We can now do some analysis of the models.

First, a large number of variables are printed out, allowing us to check out the calibration.

Second, the TROLL macro *&modeval* uses the open database to compare the value of the left-hand side of the model equations to the values of their right hand sides. They should be identical. If they are not, the incorrect value and the equation in question will be printed out. The steady-state model must work, since it was used to generate the database. A dynamic model equation will not work if its code is inconsistent with its steady-state analogue equation. Remember though, just because the dynamic model passes this test, it does not guarantee the model is correct; it simply guarantees the

dynamic model is the same as the steady-state model (and, potentially, as wrong as the steady-state model).

Finally, using the TROLL commands “simulate roots” and “lkroots”, the eigenvalues of the model are computed and printed out (for those eigenvalues between 0.9 and 1.1). You should check to see

a) the highest eigenvalue with a modulus less than one – it will tell you how long it will take for the dynamic model to converge to the steady-state model
b) that the number of leads matches the number of eigenvalues with a modulus greater than one. If the number of leads is less than those roots with modulus greater than one, then any simulation of the dynamic model will probably explode instead of converging to the steady state. If the number of leads is greater than the number of eigenvalues with a modulus of one (and you have no unit roots present), the model may have a multiplicity of solutions.

9. Run Shocks to the Dynamic Model

This section uses the TROLL macro *runshocks.src* to first create tables of the baseline steady state, and then does any shocks on the dynamic model the user wishes to do. *runshocks.src* and the files it uses, are explained in full in the following section.

The TROLL macro RUNSHOCKS.SRC

Runshocks.src consists of two major parts –

1) the main() procedure prepares the user's input about the shock to run, specifies a new set of nominal and real adjustment costs if the user wishes, runs the shock, and runs the appropriate tables and graphs using the FAME procedure *graphs.inp*.
2) procedures that are the shocks to be run on GIMF, whether they be simple impulse responses or complex agglomerations of shocks. These all call the TROLL simulation macro *simshocks.src*.

The procedure is called as follows:

&runshocks gimf1 none basecase nl 2001a 2300a 1 US atstar;

Where:
&runshocks is the name of the macro;
gimf1 is the model to be shocked;
none specifies what type of nominal rigidities and real adjustment costs are to be used
basecase is the subdirectory where the solution will be located. The graphs and tables will be in the ***basecase***\ under the subdirectory associated with the shock name and the country (in this case, *ATSTARUS*\). A further subdirectory *data*\ will contain five databases for each shock (see the section on *graphs.inp*, below);
nl is the solution algorithm to be used – either *nl*, *nonlin* or *nonlinear* to solve the nonlinear model directly, or *lin*, *lin2* or *linear* to linearise the model first and then solve it. (*lin2* uses a linearised model already present in the current working directory);
2001a 2300a are the start and end dates of the simulation respectively;
1 is the factor used to numerically linearise the model (see the end of this section on *&runshocks*);
US is the two-letter abbreviation for the country or region where the shock occurs;
atstar is the name of the subprocedure in *&runshocks* that contains the shock(s) you want to run.

You can also run two shocks consecutively, one stacked on the other:

&runshocks gimf1 none basecase nl 2001a 2300a 1 US atstar eint;

where the U.S. tradable sector productivity shock will be followed by a temporary shock to the interest rate. The resulting subdirectory created in *basecase*\ will be *atstaruseintus*\.

The procedure main()

This is an automatic procedure, that requires no user intervention; this section is for information only.

1. Some information about the shock is saved to the TROLL-specific database, the SAVE database, and the country names are retrieved.
2. Code outlining different options for the nominal and real adjustment costs of GIMF relative to the baseline calibration (i.e. double the nominal rigidities, one-half the nominal rigidities, etc.). The user can add groupings if they wish.
3. The shock (identified as *newshock*) is run. *&(newshock)* is the appropriate procedure from the section after the main procedure. The *&timesecs* macro on either side of the shock turns the timer of the shock on and off. Once the shock runs, the elapsed time of the shock will be printed out..
4. If the model is being solved using numeric linearization (explained below), the next section will appropriately transform the simulation data.
5. The FAME input file *graphs.inp* is run in FAME. It is explained further in its own section.

A typical shock subprocedure – ATSTAR()

The subprocedure is named to reflect the shock it is performing. First, you specify a title for the shock to be used in the graphs and tables using *shocktitle*:

>> do shk_shocktitle = .SCTRY.||": Permanent 2% Increase in Tradable Sector Productivity";

where *.SCTRY.* is variable representing the country you called in your &runshocks call, the United States.

You set the shock using the TROLL macro *&simeq*, which needs the date range and the expression for the shock, coded either as a number:

*≃ >> 2001a to 2300a "shk_atstar_us = con_atstar_us(-1)*1.01"*

or as an equation:

*≃ >> 2001a to 2300a "shk_atstar_us = con_atstar_us(-1)**0.60 *(con_atstar_us*(1+0.01/factor))**(1-0.60)"*

where factor is for numeric linearization (see the next section for an explanation).

You can also use mathematical expressions and/or variables for dates, provided they are enclosed in quotation marks:

*≃ >> 2001a to "2001a+299" "shk_atstar_us = con_atstar_us(-1)*1.01"*
*≃ >> "&(sdate)" to "&(sdate)+299" "shk_atstar_us = con_atstar_us(-1)*1.01"*

The procedure will then simulate the shock by calling on the simulation subprocedure *&simshock*, where the call for *&simshock* in ATSTAR() is following:

& simshock; >> 1 PERMANENT atstar_&(cc);

Where:
& simshock calls the subprocedure for solving the model using nonlinear methods;
>> passes the appropriate arguments to *& simshock* (this is called the queue-input statement);
1 is the number of pieces you want to run the shock in. In this case, a one percent productivity shock with be run as 1% shock. If it fails, it will try run it in 2 steps, each step as a 0.5% shocks, where the second step uses the results of the first step as its starting point. If that fails it will run it in 4 steps of 0.25% each, and so on. If you specified the number ***2*** instead, it will only try to run the shock as in two steps, each step being a 0.5%;
PERMANENT is the start of the list of permanent shocks you are running – this can be either "none", or in this case we are running the shock ***atstar_&(cc)***, which is a permanent productivity shock. This code is only relevant if you are running the linearised model.

A Note on Numeric Linearization

Numeric linearization is where you linearise the nonlinear model in the neighbourhood of the steady-state numerically, rather than writing a fully linear version of the model code.

You specify the numeric linearization factor as a number that is large enough to guarantee the model is fully linear in its solution. For example, if you pick a numeric linearization factor of 100, a factor of 101 or of 99 should give the exact same solution.

It executes as follows:
1) The database variable factor is created.
2) When specifying the shock using &simeq, it should be divided by factor. i.e.

*≃ >> 2001a to 2300a "shk_atstar_us = con_atstar_us(-1)*1.05"*

should be written as:

≃ >> 2001a to 2300a "shk_atstar_us = con_atstar_us(-1)(1+0.05/factor)"*

3) Once the model is simulated using shock/factor, the shock minus control answer of the shock is multiplied by factor, and added back onto the control database.

Therefore, a 1% tradables productivity shock with a numeric linearization factor will be simulated as a 0.01% shock, and the results will be multiplied by 100, to give a numeric approximation of the full 1% tradables productivity shock.

The FAME procedure GRAPHS.INP

Graphs.inp runs all the graphs and tables in FAME for the model. It is an automatic procedure, that requires no user intervention; this section is for information only.

Graphs.inp relies on the group of FAME procedures in *graphs.pro* to modify databases, table the steady state, and graph the results. In our example so far, we ran a tradables sector productivity shock on the United States, in the subdirectory *basecase*, under which a new subdirectory *ATSTARUS* will created , along with *data*

The output in *basecase\data* will be five databases:

1. *atstar_us_none.db* – Shock database (corrected for the numeric linearization factor, if any was used in the simulation).
2. *atstar_us_none_con.db* – Control database (corrected for the numeric linearization factor, if any was used in the simulation).
3. *atstar_us_none_ss.db* – Control steady-state database (the control database for the correction of the numeric linearization).
4. *atstar_us_none_o.db* – Shock database (as simulated - i.e. divided by numeric linearization factor).
5. *atstar_us_none_cono.db* – Control database (as simulated - i.e. divided by numeric linearization factor) .

The output in basecase\atstarus will be:

1. *reportss.ps* – reports the steady state of the model after the shock (for temporary shocks, it should be unchanged). Includes the national accounts, the trade matrix, and the model parameterization.
2. *tables.ps* – reports the short-run dynamics of the model after the shock, relative to the control case. Includes the national accounts, assets and debt, monetary policy, and the fiscal, corporate and financial sectors.
3. *18 individual pages of graphs* – all numbered and named, for each country, showing most of the major variables in GIMF.
4. *fullpack_US.ps* and *fullpack_CA.ps*– all 18 pages of graphs in one package for each country.
5. *financesummary.ps* – there are two pages of graphs per country; a survey page and a page of financial / corporate sector variables.
6. *the subdirectory PDF* – contains certain individual graph pages replicated as PDF files, for each country.

The TROLL procedure SIMSHOCKS.SRC

This procedure is a group of simulation procedures for solving GIMF. *No user intervention is ever required here*. This section is simply for information.

&simshock can simulate the nonlinear model (either in full, or using a numeric linearization technique) or the linearised model (using the technique found in DYNARE). It will use one of three procedures to run the model:

1. ***stepsimulate*** if the call to *&simshock* specifies "*1*", and the call to *&runshocks* specified "*nl*", "*nonlin*", or "*nonlinear*".
2. ***stepshock*** if the call to *&simshock* specifies a number greater than one, and the call to *&runshocks* specified "*nl*", "*nonlin*", or "*nonlinear*".
3. ***simulateshock*** if the call to *&simshock* specifies "*1*", and the call to *&runshocks* specified "*lin*", "*lin2*", or "*linear*".

Procedures 1 and 2 will use the numeric linearization technique, if the numeric linearization factor is greater than one. Obviously, the third procedure does not need it.

The procedure stepshock()

This subprocedure is the algorithm that allows large shocks to be split into steps and run consecutively and cumulatively to get the final answer. The number of steps is specified by the user. It is a more robust method to solve highly nonlinear models, that is using the divide-and-conquer (DAC) algorithm.

The procedure stepsimulate()

This subprocedure is similar to stepshock. It is invoked when the number of steps specified is "1". It, too, uses the DAC algorithm. If the simulation fails, it will re-try it with 2 steps (50% of the intended shock). It will split it to steps as small as 1.5625% of the original shock. It will save every step that works, and will print out whatever fraction of the shock worked through the normal graphics package. The title of the shock will be altered to reflect this. For example, the title:
United States: 1% Permanent Increase in Tradables Productivity
will become:
United States: 1% Permanent Increase in Tradables Productivity (46.875% of the shock)

The procedure simulateshock()

This subprocedure is the algorithm that linearises the nonlinear model using the linearization routine also found in DYNARE. In other words, all the endogenous variables are rewritten as functions of the lagged state variables and the contemporaneous exogenous variables. For permanent shocks, the model is linearised around the new steady-state, not that of the control case.