FRB/US Simulation Basics

June 2015

This note covers some basic information needed to set up and execute simulations with the FRB/US model. Reference is made to four example programs:

- *example1.prg* runs a simple simulation under backward-looking VAR expectations;
- *example2.prg* runs a simple simulation under model-consistent (MC) expectations;
- *example3.prg* illustrates how to run a simulation in which the outcomes for the federal funds rate are constrained by the zero lower bound (ZLB) and also by threshold conditions that may delay the rise of the funds rate from the ZLB;
- *example4.prg* runs a simulation under MC expectations using a monetary policy rule that is not one of the policy alternatives included in FRB/US and that requires the addition of new MC expectations variables to FRB/US.

The example programs liberally use EViews string variables to define the names of key inputs (eg, models, databases, simulation ranges) so that the programs can be modified easily to work with alternative inputs.

1. Expectations overview

FRB/US contains about thirty expectations terms, each of which is coded as a separate variable with its own equation. The name of each expectations variable starts with a *Z*: for example, *zpicxfe* is the expectation of next quarter's value of *picxfe*, which is the rate of inflation of core consumer prices. Each expectations variable can have either a VAR or MC solution. A VAR solution is based on a fixed-coefficient equation whose structure is consistent with the historical interrelationships among a small set of key macro variables and whose explanatory variables are observed contemporaneously or with a lag. In an MC solution, the expectation is required to be the same as the solved future value of the underlying variable. The MC equation for *zpicxfe* is simply *zpicxfe = picxfe(+1)*. The design of the FRB/US simulation programs permits some Z variables to have VAR solutions and others to have MC solutions.

EViews uses the Fair-Taylor (FT) algorithm to solve models with future-dated variables, but this method does not reliably or efficiently solve FRB/US when any of its expectations are MC. As a result, two more powerful MC algorithms have been specifically developed for EViews simulations of models with MC expectations; they are coded as EViews subroutines.[1] Because the inputs to these alternative algorithms are quite different from the inputs to the built-in EViews model *solve* procedure, the requirements of a FRB/US simulation with one or more MC expectations differ from those of a simulation in which all expectations are VAR-based. Specifically, the MC algorithms require two models, one containing the full model with all expectations formed using their backward-looking equations and the other consisting of only the forward-looking equations for the set of MC expectations variables and their errors. The

[1] See Flint Brayton, "Two Practical Algorithms for Solving Rational Expectations Models," Finance and Economics Discussion Series 2011-44. (http://www.federalreserve.gov/pubs/feds/2011/201144/201144pap.pdf)

algorithms alternate between solutions of each model, adjusting the add factors (constant adjustments) on the backward-looking equations for those expectations that are MC after each pair of simulations, until the MC expectations errors are zero. The default *newton* algorithm uses a Newton approach to calculating the constant adjustments. The alternative *qnewton* algorithm uses a version of Broyden's quasi-Newton method.

2. Subroutines and add-ins

The *subs* subdirectory of the FRB/US software package contains two libraries of subroutines. The set of subroutines in *master_libary.prg* must be made available to all simulations programs using the *include* command. The subroutines in *mce_solve_library.prg* must be made available to all programs that run simulations with MC expectations.

The FRB/US simulation programs use special add-in commands to load equations and coefficients and for some other tasks. The add-ins must be installed as described in the README.TXT file.

3. Equations and coefficients

FRB/US equations and coefficients are stored in text files in the *mods* subdirectory. The files named *stdver_coeffs.txt* and *stdver_eqs.txt* contain the coefficients and equations for the version of FRB/US in which all expectations are VAR-based. The files named *pfver_coeffs.txt* and *pfver_eqs.txt* contain the coefficients and equations for the version of FRB/US in which all expectations are MC. Only the equations for the roughly thirty expectations variables differ between the two versions.

VAR expectations (example1.prg*)*

A VAR-expectations simulation can use the add-in commands *ld_frbus_eqs* and *ld_frbus_cfs* to load all equations and coefficients.

```
ld_frbus_eqs(modelname=%varmod,modelpath=%varpath)
ld_frbus_cfs(modelname=%varmod,modelpath=%varpath)
```

The strings *%varmod* and *%varpath* need to contain respectively the name of the VAR-expectations version of FRB/US (*stdver*) and its location.

MC expectations (example2.prg)

The following code illustrates how to set up the two required models using the *mce_load_frbus* subroutine.[2]

```
%varmod = "stdver"
%varpath = "../mods/"
%mcemod = "pfver"
%mcepath = "../mods/"
```

[2] The subroutine uses four add-in commands: *ld_frbus_eqs, ld_frbus_cfs, ld_mce_eqs* and *ld_mce_cfs*. Additional documentation of the subroutine can be found in section 7 of the *MCE Solve Users' Guide.*

```
%zvars = "zdivgr zgap05 zgap10 zgap30 zrff5 zrff10 zrff30 zpi10 zpi10f zpic30
zpib5 zpic58"


call mce_load_frbus("mce_vars=%zvars,mod_b=%varmod,path_b=%varpath,
mod_f=%mcemod,path_f=%mcepath")
```

The *mce_vars* argument specifies the names of the expectations variables that are to have MC solutions. In the example, the list is given by *%zvars* and consists of the expectations variables that appear in equations for asset prices. The *mod_b* argument specifies the name of the full model in which all expectations variables have backward-looking equations, which is FRB/US version *stdver*. The *mod_f* argument specifies the name of the FRB/US version in which the forward-looking identities for the MC variables are found, which is FRB/US version *pfver*. The subroutine processes *pfver* so that it contains only the needed forward-looking identities, replaces the "Z" at the start of each variable name with a "W" in order to distinguish the expectations solutions of *stdver* from those of *pfver*, and adds an expectations error ("E") equation for each MC expectations variable.

4. Data

The *data* subdirectory contains the *longbase* database, which is in EViews format and contains historical observations on all FRB/US variables and projected observations that extend about 100 years in the future. For the first few years of the projection the values of a few key macro variables are based on the most recent "Economic Projections of Federal Reserve Board Members and Federal Reserve Bank Presidents" (SEP). Thereafter, the projections gradually converge to an illustrative but arbitrary steady-state growth path.[3] In the example programs, the string *%dbin* declares the database name.

As mentioned in the previous section, simulations with MCE expectations require the use of a second model whose equations contain special "W" and "E" variables. These variables are not part of the input database. In *example2.prg*, the *make_frbus_mcevars* subroutine is executed to create data for these variables. It sets the "W" variables equal to the "Z" variables and the "E" variables to zero.

5. Monetary policy

There are seven basic options for setting the federal funds rate. The options consist of five different policy reaction functions and policies that hold the nominal funds rate or the real funds rate at a pre-determined path. A specific option is chosen by setting one of an associated set of seven exogenous switch variables to 1 and the others to 0. The follow table indicates the names of the switches, their descriptions, and for the options associated with policy rules, the name of the associated rule equation. For more information see the HTML reference document on *FRBUS Equations*.

Switch variable	Description	Reaction function equation

[3] The initial parts of the projections of real GDP growth, core and overall PCE price inflation, the unemployment rate and the federal funds rate are based on the SEP. All other aspects of the projections included in the database should not be construed as reflecting the views of the FOMC or any of its participants.

dmpintay	inertial Taylor rule	rffintay
dmptay	Taylor rule	rfftay
dmptlr	Taylor rule with unemployment gap	rfftlr
dmpalt	estimated rule	rffalt
dmpgen	generalized rule	rffgen
dmpex	exogenous nominal funds rate	na
dmprr	exogenous real funds rate	na

The values of the switch variables can be specified directly or via the subroutine *set_mp*. For example, the command

```
call set_mp("dmpintay")
```

selects the inertial Taylor rule by setting *dmpintay*=1 and all other switches to 0. The subroutine obeys the current workfile sample, so it is possible, for example, to set up a simulation in which the federal funds rate is exogenous initially and then switches to one of the policy rules by executing a pair of calls to the subroutine for different sample periods.

The stability of FRB/US solutions over the lengthy simulation periods needed when expectations are MC requires that one of the endogenous policy rules be in force from some point not too far from the start of the simulation through its end.

The ZLB and thresholds (example3.prg)

The chosen basic option can be modified so that the outcome for the federal funds rate is subject to the zero lower bound (ZLB). Setting the exogenous variable *rffmin* to zero (or, more realistically, a small positive value) imposes the ZLB. Setting *rffmin* to a large negative number eliminates the constraint.

In addition, the timing of the liftoff of the federal funds rate from the ZLB can be determined by a version of the threshold criteria that appear in FOMC statements from December 2012 to January 2014. In this case, liftoff is delayed until either the unemployment rate falls below a threshold (given by the value of *lurtrsh*) or the expected inflation rate rises above a threshold (given by the value of *pitrsh*), when the exogenous switch variable *dmptrsh* = 1. Once either threshold is crossed, the endogenous switch variable *dmptr* becomes 1.0 and the policy rate is determined, with a one-quarter delay and subject to the ZLB, by the chosen policy option, and it continues to be set according to that option irrespective of the subsequent outcomes for unemployment and expected inflation.

Care needs to be exercised in the use of either the ZLB or the liftoff thresholds in simulations that impose baseline-tracking add factors. If the add factors on various key equations in the monetary sector are not zero (including the equations for *rffe, rffrule, dmptlur, dmptpi, dmptmax*, and *dmptr*), the effects of the zero bound restriction and the threshold conditions may not be those intended. Moreover, because the equations for *dmptlur, dmptpi, dmptmax,* and *dmptr* are designed to restrict their values to the [0,1] interval, these equations should never have non-zero add factors.

The structure of the endogenous switch variable *dmptr* requires further comment. In the equation, *dmptr* equals to the maximum of its own lag, a switch variable (*dmptlur*) indicating whether the unemployment rate has fallen below its threshold, and a switch variable (*dmptpi*) indicating whether expected inflation has risen above its threshold. The inclusion of the own lag has the effect of making the threshold restrictions a one-time event, but it requires that the value of *dmptr* be zero in the quarter prior to one in which it is desired to have threshold conditions affect the solution value of the funds rate. If this quarter is the first simulation period, then it is simply a matter of setting the baseline data on *dmptr* to zero in the quarter prior to the start of the simulation. Alternatively, in a simulation in which, for example, the unemployment rate is below its threshold initially and one wants to impose the threshold conditions only after the labor market weakens sufficiently, then the desired condition for *dmptr* also requires that the initial values of threshold variable *lurtrsh* (and perhaps *pitrsh* as well) be set to extreme values --- such as -9999 for *lurstsh* and +9999 for *pitrsh*. This is illustrated in *example3.prg.*

Equilibrium real federal funds rate for monetary policy reaction functions (rstar)

The monetary policy equations for *rffintay, rfftay, rfftlr,* and *rffgen* contain the variable *rstar*, which can be interpreted as the policymakers' estimate of the equilibrium real funds rate. The behavior of *rstar* is controlled by the exogenous switch variable *drstar*. When *drstar =0*, *rstar* is exogenous; when *drstar=1*, *rstar* moves gradually toward the actual real funds rate. As a general matter, the endogenous setting is needed for simulations of any permanent shock that changes the equilibrium rate of interest, which in FRB/US includes most types of permanent shocks. For other shocks, either setting is acceptable, but outcomes may be sensitive which alternative is chosen, especially under MC expectations.

6. Fiscal policy

The exogenous switch variables *dfpex, dfpsrp,* and *dfpdbt* can be specified directly or via the subroutine *set_fp* to select one of three possible settings for the behavior of the trend federal and state and local personal income tax rates, *trfpt* and *trspt*. When *dfpex* = 1.0 and the other switches are zero, the trend tax rates are exogenous; when *dfpsrp* = 1.0, the trends adjust to gradually stabilize the ratios of the two government surpluses to GDP at the values given by exogenous variables *gfsrt* and *gssrt*; when *dfpdbt* = 1.0, the trends adjust to gradually stabilize the debt-to-GDP ratios of the each government sector at the values given by the exogenous variables *gfdrt* and *gsdrt*.

The syntax for use of the subroutine is *set_fp("dfp…")*. For example,

```
call set_fp("dfpsrp")
```

selects surplus stabilizing fiscal policy by setting *dfpex*=0*, dfpsrp*=1*,* and *dfpdbt*=0. The subroutine obeys the current workfile sample, so it is possible, for example, to set up a simulation in which fiscal policy is exogenous initially and then switches to surplus targeting by executing a pair of calls to the subroutine for different sample periods.

The lengthy simulation periods associated with FRB/US simulations with MC expectations usually require that one of the endogenous fiscal policy options be in force from some point not too far from the start of the simulation through its end.

7. Add factors and baseline tracking

Each of the example programs sets up FRB/US so that all of its equations are assigned add factor variables (*addassign*) and their values are initialized (*addinit*) such that, in the absence of shocks, the simulated outcomes for all endogenous variables replicate the baseline database. Because the supplied databases do not generally impose the various MC identities, the expectations in MC simulations with baseline tracking are MC only when expressed as deviations from baseline.

The EViews *addassign* command creates an add factor variable for each equation whose name is constructed by adding the suffix *_a* to the name of dependent variable. Each FRB/US equation also comes coded with a separate add factor variable whose name contains *_aerr* as a suffix. If the *_aerr* variables do not have values in the input database, they need to be given values (typically zero) in the simulation program.

8. Simulation

The default EViews model *solve* options are typically not precise enough for satisfactory solutions of FRB/US. The following command specifies a better choice of options.

```
{%varmod}.solveopt(o=b,g=12,z=1e-12)
```

(The string *%varmod* refers to a previously defined name of a FRB/US version.) FRB/US simulations that use the EViews quasi-Newton Broyden algorithm (*o=b*) usually converge successfully, although some of the equations in FRB/US contain functions with kinks, which is a situation which can be problematic for Newton-type solution algorithms. When convergence fails, especially if a "solver stalled" error message occurs, switching to the EViews Newton (*o=n*) or Gauss-Seidel (*o=g*) algorithms may work.

As noted earlier, the mechanics of running a FRB/US simulation with VAR expectations are different from those associated with MC expectations.

VAR expectations

Because of the assumption of VAR expectations in *example1.prg*, the code for simulating the effects of a one-time 100 basis point upward shock to the chosen monetary policy rule (*rffintay)* can use the EViews model *solve* procedure and thus is quite simple.

```
smpl %simstart %simstart
rffintay_aerr = rffintay_aerr + 1
smpl %simstart %simend
{%varmod}.solve
```

The shock is applied to *rffintay_aerr*, but it could have been applied to *rffintay_a* with the same effect.

MC expectations

The simulation code in *example2.prg* requires a call to the *mce_run* subroutine, instead of the EViews *solve* command, because of the assumption in this case that some expectations are MC.

```
smpl %simstart %simstart
rffintay_aerr = rffintay_aerr + 1
%modstr = "mod_b=%varmod,mod_f=%mcemod,mce_vars=%zvars"
%algstr = "meth=qnewton"
%simstr = "type=single"
smpl %simstart %simend
call mce_run(%modstr,%algstr,%simstr)
```

A comprehensive description of the syntax and uses of the *mce_run* subroutine is contained in the reference document *MCE Solve Users' Guide.* The subroutine has three arguments, each of which is a string. The first string (%modstr) contains information about the model, which can be organized in several different ways. The example uses a syntax that requires the name of the full model with VAR expectations (*mod_b* keyword), the name of the partial model with MC expectations (*mod_f* keyword), and the names of expectations variables that are MC (*mce_vars* keyword). The second string (*%algstr*) contains information about the MC solution algorithm. The example selects the quasi-Newton algorithm, which is usually the fastest of the available algorithms when running a single FRB/US simulation. The third string (*%simstr*) selects the type of simulation. Possible types include single (*single*) and optimal control (*opt*, *opttc*) simulations. The three example programs execute *single* simulations. The program named *ocpolicy.prg* illustrates how to set up and run an optimal control program of the *opt* type.

Any FRB/US simulation with MC expectations requires that the simulation period be long enough that the solved values over the period of interest are unaffected by the adding an additional quarter to the simulation period. For a simulation of a transitory shock in which the period of interest is the first five years (20 quarters) of the simulation, a range of 50 years (200 quarters) is usually sufficient. Simulations of permanent shifts, such as a change in the target rate of inflation or the target ratio of federal debt to GDP, may require longer simulation periods, depending on how various terminal conditions are set, as described in the next paragraph.

A FRB/US simulation with MC expectations also requires terminal conditions for all variables whose future values appear in the partial model with the MC identities. For the example equation noted earlier, *zpicxfe = picxfe(+1)*, the value of *picxfe* must be defined in the first quarter after the end of the simulation. Some of the other MC identities contain as many as four leads. For simulations that do not affect the long-run values of the specific variables having leads, the values of these variables in the input database should provide satisfactory terminal conditions, as long as the simulation range ends a sufficient number of quarters before the end of the input database. For a permanent shift, adjusting the baseline values of terminal data for the likely effects of the shift can significantly shorten the required length of the simulation period. Adjustments of this type are done automatically when the *%simstr* argument to the *mce_run* subroutine contains the keyword *terminal*. When this option is invoked, terminal values are adjusted in the first simulation iteration based on the solution values of the version of the model with VAR expectations. This procedure works satisfactorily, because FRB/US has essentially the same long-run properties under VAR expectations as it does under MC expectations.

9. Modifying the structure of FRB/US

There are two approaches to making changes to FRB/US equations. In one approach, changes are made after the standard add-ins described above (*ld_frbus_eqs* and *ld_frbus_cfs*, or *mce_load_frbus*) have been used to load the model into the workfile. This approach applies to modifications to an equation's coefficient values, which can be entered directly in the appropriate coefficient vector after it has been loaded (eg, y_rfftay(3) = 2). This approach can also be used for changes to the functional forms of equations when running EViews 8, which has the built-in *replace*, *append*, *merge*, and *drop* procedures for working with the equations of a model.

In the second approach to making changes to FRB/US equations, the add-in commands supplied in the FRB/US Model Package are used to load only those equations that do not need modifications. Then, additional EViews commands are used to "append" or "merge" the new specifications of the remaining equations. This approach must be used when making changes to functional forms in EViews 7 and may be optionally used for this purpose in EViews 8. For a simulation under VAR expectations, the *ld_some_eqs* add-in loads a subset of FRB/US equations. With this add-in, only those equations whose names match or do not match the names in the string assigned to the *eqnames* keyword are loaded. If the first "word" in the string is "allbut", then all equations but those listed are loaded. Otherwise, only those equations listed are loaded.

```
%eqs = "allbut rffe"
ld_some_eqs(modelname=%varmod,modelpath=%varpath,eqnames=%eqs)
{%varmod}.append rffe = <an alternative text specification>
```

In this example, all equations but the one for RFFE are loaded and an alternative text specification for RFFE is appended. (Because of the ease of changing coefficients after they have been loaded, there is no need to have an add-in that loads a subset of coefficients.)

In the case of a simulation with model-consistent expectations, the second approach to making changes to functional forms uses the optional "allbut" and "only" keywords of the *mce_load_frbus* add-in to load a subset of equations. The following example, which is taken from *example4.prg*, loads all but the RFFGEN equation and then replaces it with a new specification.

```
%eqs = "rffgen"
call  mce_load_frbus("mce_vars=%zvars,mod_b=%varmod,path_b=%varpath, _
      mod_f=%mcemod,path_f=%mcepath,allbut=%eqs")
{%varmod}.append rffgen = <an alternative text specification>
```

Modifications to FRB/US that introduce new MC expectations variables need special attention to ensure that they conform to the design of the MCE algorithms supplied in the FRB/US Model Package. As mentioned earlier, these algorithms require two models, one containing the full set of FRB/US equations in which all expectations are coded using backward-looking identities, and one with a partial set of equations consisting of the forward-looking MCE identities and an associated set of identities that defines expectations errors as the difference between the forward-and backward-looking solutions. In

this framework, each MCE variable is associated with three equations and three variable names. The name of an expectations variable when it appears as the dependent variable in its backward-looking equation in the first model (and when it appears as an explanatory variable in other equations in that model) should start with the letter "z". Its name when it appears as the dependent variable in its forward-looking equation should replace the initial "z" with a "w". And the name of its expectations error should be formed by appending an "e" to the "z" form of its name. See *example4.prg* for a concrete example of how to add new MCE variables.