Recursively Running Dynare Mod File

Xiangyang Li*

March 13, 2016

In this section, I am going to cover the topic that how we can run the mod files recursively. Recursively running of a mod file is necessary sometime. For example, if you want do some sensitivity tests of a parameter, and you want to see how a small changes in this parameter will affect the desired results of your model or questions.

1 The Basic Logic of Recursively Running

Usually, we want to test how one or more structural parameters of the model change have any consequences. These changes will usually be small and continuous. For example, if we want to study the changes of the elasticity of substitution of two different inputs (η) will has any consequences on output, then we may set η equals to array of values like $\eta = 1.50 : 0.01 : 2.00$ in Matlab or any values that are reasonable. This setting is equivalent to $\eta = 1.50, 1.51, 1.52, 1.53, \cdots, 2.00$. For each η in this data array, we want run the mod file once and see any changes happen. The basic logic behind this running will involve the following steps in Matlab:

1. Initial Setup
2. Assign data array to the parameters for testing
   (a) for each parameters in data array
   (b) run the mod files
   (c) save the related results in oo_,M_ or options_ objects into a mat file
   (d) end iteration
3. Loading the saved mat file and do desired analysis.

*The Note is still incomplete and preliminary. Typos and Errors may still prevail. This is the first draft and improvements are expected. Any comments and suggestions are welcomed! Please send to ahnulxy#qq.com (Replace # with @ when email). This series notes dedicate to help students and researchers to understand more about Dynare and DSGE models. Special thanks goes to Prof. Christiano for his inspiring lectures and notes.

There are few points that need further explanations:

1. You need write a extra m file to implement this algorithm or idea and then invoke the mod file using dynare command.

2. Most of the results of the solution and estimation are stored into three objects (or structs in Matlab) by Dynare automatically: oo_, M_ and options_ after each running. The objects you declared to store your results will be cleared if you do not put 'noclearall' option after the 'dynare' command. The three objects will be overwritten each iteration regardless of the option is specified or not.

3. You can extract the desired data from this objects and then store into a Matlab mat file. After you finish the iteration, you can load the mat file and do whatever you want to do.

The mechanism for recursively invoking is simple and could be easily adopted to your codes. The key point is to pass the parameters into the mod file or complied m file by loading the saved mat file into mod file. The saved mat file is updated or replaced in every loop so as to pass new parameters into mod file.

2 An Illustration

We use a simple RBC model to illustrate the whole thing. In this example, we just carry out a simple experiment to see how the change of parameter α, the capital share in production function will has any effects on output. I will first list the mod file and then the Matlab code to invoke the mod file recursively.

This is the mod file[1]:

```
%This program is written by Xiangyang Li @UND
%This is a very simple RBC model;

var y i r n w k a R c;
varexo e;
parameters alpha beta delta rho sigma psi eta sigmae;
parameters yss iss rss nss wss kss Rss css;

%load the parameters saved in the mat file;
%load is the matlab command which can be used here;
load parametersaved;

%this function will pass the saved parameter into this mod file
%set_param_value() is the built-in function of dynare.
set_param_value('alpha',alpha);
```

[1]see recursively_running_dynare_mod.

```
beta = .99;
delta = .02;
rho = .97;
sigma = 1;
psi =3;
eta = 1;
sigmae = .01;

rss = 1/beta - 1;
Rss= 1/beta - 1 + delta;
kn = (alpha/Rss)^(1/(1-alpha));
wss = (1-alpha)*kn^alpha;
nss = ((kn^alpha - delta*kn)*(psi/wss)^(1/sigma))^(-1/(1+eta/sigma));
kss = kn*nss;
yss = kn^alpha*nss;
iss=  delta*kss;
css = yss - iss;

model;
% (1) Euler equation, capital
exp(c)^(-sigma)=beta*exp(c(+1))^(-sigma)*(exp(R(+1))+(1-delta));

% (2) Euler equation, bonds
exp(c)^(-sigma)=beta*(1+r)*exp(c(+1))^(-sigma);

% (3) Labor supply
psi*exp(n)^(eta)=exp(c)^(-sigma)*exp(w);

% (4) Production func
exp(y)=exp(a)*exp(k(-1))^(alpha)*exp(n)^(1-alpha);

% (5) Capital demand
exp(R)=alpha*exp(a)*exp(k(-1))^(alpha-1)*exp(n)^(1-alpha);

% (6) Labor demand
exp(w)=(1-alpha)*exp(a)*exp(k(-1))^(alpha)*exp(n)^(-alpha);

% (7) Resource constraint
exp(y)=exp(c)+exp(i);

% (8) Capital accumulation
exp(k)=exp(i)+(1-delta)*exp(k(-1));

% (9) Productivity shock
a=rho*a(-1)+e;
end;
```

```
initval;
k=log(kss);
y=log(yss);
c=log(css);
i=log(iss);
a=0;
r=rss;
R=Rss;
w=log(wss);
n=log(nss);
end;

shocks;
var e = sigmae^2;
end;
resid(1);
steady;
check;

%everything will show up in each loop which will consume a lot of time
%stoch_simul(order =1);

%useful for loops, most info. will not display,
%but compling info. still show up.
%stoch_simul(order =1,nograph,nofunctions);
stoch_simul(order =1,noprint,nograph);
```

And then the Matlab codes[2]:

```
%This is the main file which invoke the mod file
%initial setup
clear all;
close all;
clc;

%sensitivity setting, only one parameter is set for illustration.
alpha_d = [0.3,0.35,0.4];

%where results are saved, here we are interested in the
%IRF of output y with respect to exogenous shock e, y_e;
%the default periods is 40 for IRF.
saved_results=zeros(40,length(alpha_d));

%delete old results mat file.
```

[2]see recursively_running_dynare_main.m.

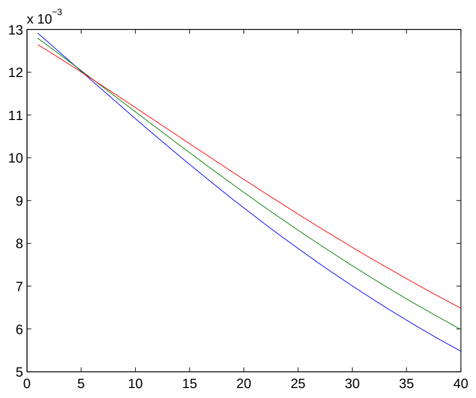
```
if exist('saved_results.mat','file')
delete saved_results.mat
end

for ii=1:length(alpha_d)
    %delete the old mat file
     if exist('parametersaved.mat','file')
    delete parametersaved.mat
     end

    %new parameter value are assigned
    alpha= alpha_d(ii);

    % alpha will be saved in a mat file named as parametersaved.mat
    % if more parameters are stored you may use the command like
    % save parametersaved alpha beta ...;
    save parametersaved alpha;

    %new parameters will be loaded in the Dynare mod file;
    %see mod file for how we load new parameters into mod file;
    dynare recursively_running_dynare noclearall

    %directly invoke the complied m file, you need
    %use the dynare command at first place.
    %recursively_running_dynare

    %IRF of output y w.r.t exogenous shock;
    saved_results(:,ii) = y_e;
end

%% Plot the IRF from the results saved.
%we are not going to detail the plot.
 plot(saved_results*100);
 legend('\alpha =0.3','\alpha =0.35','\alpha =0.4');
```

Here is the plot:

3 Efficiency Problems

In this section, I am going to cover some small tricks to help you improve the running efficiency of your iteration of the mod file. In most cases, this tricks will help you save a lot of time and computing resources.

Figure 1: How output y changes with α



3.1 Using the Compiled m file

This trick will save us from using dynare command in the Matlab codes anymore once we complied (after using dynare command) the mod file and get the correspondent m file (running this m file will be equivalent to run the mod file). We can directly invoke the m file instead of using dynare command. This means that we do not compile the mod file every time in each loop of the iteration and use the m file instead. Hence, this will save us a lot of time especially when the model is complicate.

3.2 Using noprint, nograph options

Even when use the m file in each loop, if you plot all IRF of endogenous variables, you may still have low efficiency in running your iteration. This is because Dynare will do a lot of useless jobs in each loop. This will be time-consuming and unnecessary if we only care about one or two, or even none of them but you still let Dynare plot all IRF for all endogenous variables. So we need tell dynare to suppress the output like IRF figures, screen printings if we want to save time. And most often, using options like noprint, nograph, nofunctions in stoch_simul command will save us a lot of time[3].

[3]Compiling info. will persist in showing up no matter what options you have put.