Further Practicing Dynare

Li Xiangyang*

School of Finance, SUFE

September 4, 2013

Abstract

This note comes from the understanding when I read Practicing Dynare by Barillas et al. (2010) which is one of the excellent notes I ever met. This note borrows heavily from Barillas et al. (2010) and Sims (2012) in University of Notre Dame among others. However, the mod files in Practicing Dynare are modified to run in Dynare 4.2.0 and Matlab 7.6(2008a). Furthermore, This note provides more details about the Dynare estimation results which aims to give deep and further understanding of using Dynare. At least, this note will give you some very basic ideas and intutions, such as simulations, impulse response etc. that underlying behind the estimation.

*Ph.D. candidate in SUFE, Email: ahnulxy#163.com; This note is still incomplete and preliminary and I welcome your comments and suggestion!

Contents

1 Introduction

The 9 models provided by Sargent's note illustrated how to estimate various macroeconomic model, especially the DSGE models in Dynare. This practices fully display the power of Dynare as a useful tool in Macroeconomics. The note help me to establish some of the basic ideas about using Dynare which are proved to be essential and fundamental. Due to the insufficient illustration examples or help documents to beginners provided by the official website, it has been a painful learning process for me-the beginner-to learn to use Dynare to estimate my model until I found the latest version note by Barillas et al. (2010). But unfortunately, some of the mod files do not work in my environment though those mod files are written in 2010 or perhaps early. To modify all those codes to fit the latest version of Dynare is a challenging task for me. It is time- and energy-consuming. But when you make through, it is exceptionally rewarding.

When we do not have data in hand but still want to conduct some analysis[1], then we can simulate the data for endogenous variables from the FOC equations or the log-linearized equations system. The driving forces behind come from exogenous variables, such as, the technology shock, which are commonly assumed to have a i.i.d Normal distribution. This kind of simulation is referred as stochastic simulation. In Dynare 4.2.0, we use command *stoch_simul* to conduct the simulation. However, when no exogenous shocks in the model, we might want to inspect the impact of a shock to one or more endogenous variables, such as a permanent or temporary increase in government spending or households consumption. This kind of simulation is referred as deterministic simulation. In such a case, command *simul* is used. If we have data for all or some of the endogenous variables, then we may estimate some parameters instead of calibrating all parameters as we do when data are not available. You can instruct Dynare to estimate your parameters interested by Bayesian or Maximum Likelihood methods. If we have estimated the model when data available and the parameters are estimated, then we can also use those estimated parameters and calibrated ones to do stochastic simulations using *stoch_simul* or deterministic simulations using *simul*.

[1] All parameters must all known! Otherwise, estimation must be done before stochastic simulation can carry out!

2 Stochastic Simulation

stoch_simul computes a Taylor approximation of the decision and transition functions for the model, steady state values of endogenous variables, a model summary, impulse response functions and various descriptive statistics (moments, variance decomposition, correlation and auto-correlation coefficients). There are a number of options following this command. If no options are provided, it is going to give you the default output. By default, Dynare does a second order approximation about the steady state and hence the policy and transition functions looks more complicated with cross terms. I will give no detailed options explanation, you can refer the Dynare_Team (2013).

Dynare will produce policy and transition functions for the endogenous variables of the model in state space form. Letting s_t be a $m\times 1$ vector of states and x_t be a $n\times 1$ vector of controls (here I include "static" controls like output as well as dynamic controls; you do not need to eliminate the static controls to use Dynare). The state space representation of the system can be written:

$$\begin{aligned} s_t &= As_{t-1}+B\varepsilon_t \\ x_t &= \Phi s_t \end{aligned}$$

where Φ is the policy function and a matrix of $n\times m$. Dynare writes the system out by substituting the state transition equation into the policy function as follows:

$$x_t = \Phi A s_{t-1}+\Phi B\varepsilon_t \equiv Cs_{t-1}+D\varepsilon_t$$

Then we can write out the whole system as:

$$\begin{aligned} s_t &= As_{t-1}+B\varepsilon_t \\ x_t &= Cs_{t-1}+D\varepsilon_t \end{aligned}$$

We can combine this into one expression by denoting $Y_t=[s_t,x_t]^{'},\Psi=[A,C]^{'}$and $\Omega=[B,D]^{'}$:

$$Y_t=\Psi s_{t-1}+\Omega\varepsilon_t. \tag{1}$$

To avoid any misunderstanding of the note with what the Dynare produced, further explanation is needed. The equation (1) above do not exactly correspond to what produced in Dynare. However, for an order-1 approximation, the final results of the Dynare produced has the form:

$$y_t = y^s+Ey^h_{t-1}+Fu_t \tag{2}$$

where $y_t^h = y_t - y^s|_{states}$ represents the state variables in $y_t - y^s$,[2] y_t is a column vector of endogenous variables. y^s is the steady state values of y_t[3]. The coefficients of the E matrix are stored in "oo_.dr.ghx"; the coefficients of the F matrix are stored in "oo_.dr.ghu"; and the steady state values are stored in "oo_.dr.ys" but in declaration order. However, there is one thing that deserved your attention: the order of the endogenous variables in matrix E and F are not the same as the declaration order (which is reflected in M_.endo_names). There are actually in DR-order: static variables appear first, then purely backward variables, then mixed variables, and finally purely forward variables, and within each type, the orders are in declaration order[4]. For an order-2 approximation, the results are of a little complexity:

$$y_t^h = 0.5\Delta^2 + Ey_{t-1}^h + Fu_t + 0.5G\left(y_{t-1}^h \otimes y_{t-1}^h\right) + 0.5H\left(u_t \otimes u_t\right) + I\left(y_{t-1}^h \otimes u_t\right)$$

for more detailed explanation of the coefficients, see Dynare_Team (2013).

2.1 The Model

This model is borrowed from Barillas et al. (2010).

A representative household's problem is

$$max_{\{c_t,l_t\}_{t=0}^{\infty}} E_0 \sum_{t=1}^{\infty} \beta^{t-1} \frac{(c_t^{\theta}(1-l_t)^{1-\theta})^{1-\tau}}{1-\tau} \tag{3}$$

subject to the resource constraint

$$c_t + i_t = e^{z_t} k_t^{\alpha} l_t^{1-\alpha} = y_t \tag{4}$$

the law of motion for capital

$$k_{t+1} = i_t + (1-\delta)k_t \tag{5}$$

and the stochastic process for productivity

$$z_t = \rho z_{t-1} + s\varepsilon_t \tag{6}$$

with $\varepsilon_t \sim N(0, \sigma^2)$.

The FOC of the model w.r.t consumption and labor supply as follows:

$$\frac{\left(c_t^{\theta}(1-l_t)^{1-\theta}\right)^{1-\tau}}{c_t} = \beta E_t \left(\frac{\left(c_{t+1}^{\theta}(1-l_{t+1})^{1-\theta}\right)^{1-\tau}}{c_{t+1}} \left(1 + \alpha e^{z_t} k_t^{\alpha} l_t^{1-\alpha} - \delta\right) \right) \tag{7}$$

[2] In this example, $y_t^h = s_t = [k_t, z_t]'$ noticing that the variables has alike AR(1) form.

[3] The following is my understanding of the equation (2) : if y_t is in level or log-level, then y^s corresponds to the steady state values of level or log-level; if y_t is log-linearization, i.e. percentage deviation from steady state values, then y^s should be zeros.

[4] For further explanation of DR-order and variables' definition, See the Dynare Reference Manual, Typology and ordering of variables.

$$\frac{1-\theta}{\theta}\frac{c_t}{1-l_t} = (1-\alpha)\, e^{z_t} k_t^{\alpha} l_t^{-\alpha} \tag{8}$$

An equilibrium is characterized by a system of 6 equations: (2-6) above with equation (2) has two equal signs.

2.2 Approximation

You can input the FOC equations (variables in level or in log-level) and also log-linearized equations into Dynare mod file. All are acceptable. Here we use FOC equations both in level and log-level. If log-linearized equations input, then you should change the keyword 'model;' into 'model(linear);' to tell Dynare you have input log linearization equations.[5].

We typically want to approximate the solutions of natural logs of the variables, so that impulse responses, etc. are all in percentage terms. Dynare will do linear approximation of the levels of the variables. To get it to do linear approximation of the logs of the variables, you need to specify the variables as "exp(x)". This way the variable x is interpreted as the log of the variable of interest while exp(x) is the level (since the exponential and log are inverse functions) which is what actually shows up in the most of the FOCs. Then you just type the FOCs.

In Dynare, the timing of the variable reflects the date when the variable is decided. For instance, the capital stock for time t is decided in time $t-1$(end of period). In order to exploit the syntax of Dynare, the timing for capital stock in capital accumulation equation is modified.

```
1  //GrowthApproximate.mod, variables in levels
2  var c k lab z;
3  varexo e;
4  parameters bet the del alp tau rho s;
5  bet = 0.987;
6  the = 0.357;
7  del = 0.012;
8  alp = 0.4;
9  tau = 2;
10 rho = 0.95;
11 s   = 0.007; //standard deviation of shock
12 model;
```

[5]The following is my understanding, but I am not so sure if it is correct: if the model is declared to be linear, then initval block no longer needed, the endogenous variables are automatically assigned zeroes as their steady state values since it is the essence of the log-linearization.

```
13 (c^the*(1-lab)^(1-the))^(1-tau)/c=bet*((c(+1)^the*(1-lab(+1))^(1-
     the))^(1-tau)/c(+1))*(1+alp*exp(z(+1))*k^(alp-1)*lab(+1)^(1-alp)
     -del);
14 c=the/(1-the)*(1-alp)*exp(z)*k(-1)^alp*lab^(-alp)*(1-lab);
15 k=exp(z)*k(-1)^alp*lab^(1-alp)-c+(1-del)*k(-1);
16 z=rho*z(-1)+e;
17 end;
18
19 initval;
20 k   = 1;
21 c   = 1;
22 lab = 0.3;
23 z   = 0;
24 e   = 0;
25 end;
26
27 shocks;
28 var e=s^2;
29 end;
30
31 steady;
32
33 // option simul_seed is set to ensure replication of the results
     present here.
34 stoch_simul(periods=1000,irf=40,simul_seed=123456,order=1);
35
36 dynasave('simudata.mat'); //save simulated endogenous variables, c,
     k,z,lab.
```

```
1  //GrowthApproximate_exp.mod, variables in log-level.
2  //Furthermore, we include two more variables in the mod file:
       y and i.
3  var y c k i lab z
4  varexo e
5  parameters bet the del alp tau rho s;
6  bet = 0.987;
7  the = 0.357;
8  del = 0.012;
9  alp = 0.4;
10 tau = 2;
11 rho = 0.95;
12 s   = 0.007;
13
14 model;   (exp(c)^the*(1-exp(lab))^(1-the))^(1-tau)/exp(c)=bet
     *((exp(c(+1)))^the*(1-exp(lab(+1)))^(1-the))^(1-tau)/exp(c
```

```
    (+1)))*(1+alp*exp(z(+1))*exp(k)^(alp-1)*exp(lab(+1))^(1-
    alp)-del);
15  exp(c)=the/(1-the)*(1-alp)*exp(z)*exp(k(-1))^alp*exp(lab)^(-
    alp)*(1-exp(lab));
16  exp(k)=exp(i)+(1-del)*exp(k(-1));
17  exp(y)=exp(z)*exp(k(-1))^alp*exp(lab)^(1-alp);
18  exp(y)=exp(c)+exp(i);
19  z=rho*z(-1)+e;
20  end;
21
22  initval;
23  k   = log(2.71828);
24  c   = log(2.71828);
25  lab = log(exp(0.3));
26  z   = log(1);
27  e   = log(1);
28  end;
29
30  shocks;
31  var e=s^2;
32  end;
33  steady;
34  stoch_simul(periods=1000,irf=40,simul_seed=123456,order=2); %
    if periods not specify, there will be no simulations.
35  forecast;
36  dynasave('simudata_exp.mat');
```

There is one point that you might notice in above two mod files: the two technology shocks are declared as the same forms and hence they have the same steady state values. The shock z itself has steady state value 0 and this ensures that $exp(z)$ has a steady state value of unity. The last line of the code save the simulated data (only simulated endogenous variables) in mat files. Hence, in the future, you can use the data to estimate some parameters in the model.

2.3 Approximation Result

After running above mod file[6], we have all necessary components in Matlab to conduct the further analysis. The steady state values are computed by Dynare using no-linear solver. You can substitute those steady-state values into the FOC equations and you will get identities. If you know how to compute the steady-state values, you might provide the algorithm to Dynare, for more details, see the Dynare_Team (2013).

2.3.1 Steady State Values and Model Summary

```
1  Starting Dynare (version 4.2.0).
2  Starting preprocessing of the model file ...
3  Found 4 equation(s).
4  Evaluating expressions...done
5  Computing static model derivatives:
6   - order 1
7  Computing dynamic model derivatives:
8   - order 1
9  Processing outputs ...done
10 Preprocessing completed.
11 Starting MATLAB/Octave computing.
12
13 STEADY-STATE RESULTS:
14 c                 1.49163
15 k                 29.2885
16 lab               0.291593
17 z                 0
18
19 MODEL SUMMARY
20 Number of variables:          4
21 Number of stochastic shocks: 1
22 Number of state variables:    2
23 Number of jumpers:            3
24 Number of static variables:   0
```

We totally have 6 endogenous variables $y_t, i_t, c_t, k_t, l_t, z_t$ and one exogenous variable (for log-level variables mod file), the stochastic shock e_t[7].

[6] We report the results of the two model file. This will not confuse you since the mod files have different number of endogenous variables. The first mod file has 4 while the second has 6.

[7] Jumpers should be control variables or controls, here, c_t, l_t, e_t should be controls while k_t, z_t should be state variables. Static variables are variables only appear in time t subscript. Forward-looking variables are variables appear in time $t+1$ and t subscripts. Pre-determined variables are variables appear in time $t-1$ and t subscripts. A variable can be both forward-looking and predetermined.

2.3.2 Policy and Transition function

The approximated solution of a model takes the form of a set of decision rules or transition equations expressing the current value of the endogenous variables of the model as function of the previous state of the model and shocks observed at the beginning of the period. The decision rules are stored in the structure *oo_.dr*.

For $order = 1$, the policy function reads as follows[8]:

POLICY AND TRANSITION FUNCTIONS

	c	k	lab	z
Constant	1.491626	29.288520	0.291593	0
k(−1)	0.027972	0.978257	−0.001831	0
z(−1)	0.622876	1.853488	0.191284	0.950000
e	0.655659	1.951040	0.201351	1.000000

TRANSITION FUNCTION:

$$\begin{pmatrix} k_t \\ z_t \end{pmatrix} = \begin{pmatrix} ks \\ zs \end{pmatrix} + \begin{pmatrix} 0.978257 & 1.853488 \\ 0 & 0.950000 \end{pmatrix} \begin{pmatrix} k_{t-1} - ks \\ z_{t-1} - zs \end{pmatrix} + \begin{pmatrix} 1.951040 \\ 1.000000 \end{pmatrix} e_t \quad (9)$$

POLICY FUNCTION:

$$\begin{pmatrix} c_t \\ l_t \end{pmatrix} = \begin{pmatrix} cs \\ ls \end{pmatrix} + \begin{pmatrix} 0.027972 & 0.622876 \\ -0.001831 & 0.191284 \end{pmatrix} \begin{pmatrix} k_{t-1} - ks \\ z_{t-1} - zs \end{pmatrix} + \begin{pmatrix} 0.655659 \\ 0.201351 \end{pmatrix} e_t \quad (10)$$

where $ks = 29.2885, zs = 0, cs = 1.491626, ls = 0.291593$ represent the steady-state values of correspondent variables.

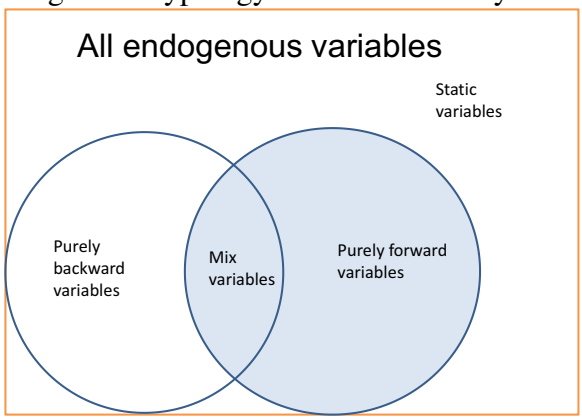
2.3.3 Simulated Endogenous Variables

We first show how technology variable z_t be simulated in the system. Almost all simulated results are stored in object or struct *oo_*. This name seems to be very strange for beginners. But you have to accustomed to it. Specifically, the simulated exogenous variables and endogenous variables are stored in *oo_.exo_simul* and *oo_.endo_simul* respectively. In our case, *oo_.exo_simul* contains a 1000*1 vector which corresponds to the only exogenous variable, e_t while *oo_.endo_simul* contains a 4*1000 matrix whose rows correspond to four endogenous variables c_t, k_t, l_t, z_t in order as the one we declared in the model using *var* command, i.e. declaration order, and we will discuss more later[9]. The following is the Matlab command we used to obtain the simulated z_t using the definition which are compared to what we get

[8]The following transition and policy functions based the results from level data mod file.

[9]Option 'periods=1000' must not omit from the command. The default value for periods is zero.

Figure 1: Typology of Variables in Dynare



by Dynare. Only the first 10 periods are compared for simplicity. The other endogenous variables are simulated in the similar way.

Sometimes, we want to do our own simulation. Hence, it is helpful to know the state space representation of the model. In term of the notations introduced above, In our example (GrowthApproximate_exp.mod), the declaration is in order: output, consumption, investment, capital stock, labor and technology shock:y, c, k, i, lab, z. The static variables are output y and investment i, the purely backward variables are capital stock k and technology shock z, and finally is the purely forward variables consumption c and labor l. And there are no mixed variables. Hence, the DR-order is y, i, k, z, c, l.[10]

Listing 1: Simulation for technology variable z_t

```
ee=oo_.exo_simul(1:10,1);
zz=zeros(10,1);
zz(1,1)=rho*0+ee(1,1); %the first value;
for t=2:10
    zz(t,1)=rho*z(t-1,1)+ee(t,1);
end
%the 6th row is the technology variable z
[zz,oo_.endo_simul(6,1:10)']
```

Listing 2: Simulation for all endogenous variables

```
//simulation code can be found in GrowthApproximate_exp_simulation.m
```

[10]Use M_.endo_names to see the declaration order and M_.endo_names(oo_.dr.order_var,:) to see DR-order.

```
n=4; %number of controls
m=2;%number of states

psi=oo_.dr.ghx;
omega=oo_.dr.ghu;

%%declaration order to DR-order,matrix indices reference.
ss=oo_.dr.ys(oo_.dr.order_var);
T=200; %number of periods for simulation
es=oo_.exo_simul(:,1);
Xsim=zeros(n+m,T);
Xsim(:,1)=omega*es(1,1);

%%Xsim already in log_difference, i.e. percentage deviation
for j=2:T
        Xsim(:,j)=psi*[Xsim(3:4,j-1)]+omega*es(j,1);
end
%%log_difference = log_level -  log_steadyvalues
for j=1:T
        Xsim(:,j)=Xsim(:,j)+ss;
end
%%The following three columns are the same.
%%oo_.endo_simul are in declaration order.
%%Xsim are in DR-order
[zz oo_.endo_simul(6,1:100)' Xsim(4,1:100)']
```

Dynare will also directly do simulations for you if you provide proper options[11]. The simulation have some of the features we might expect - consumption is "smoother" than output, which is less volatile than investment. The capital stock evolves very smoothly as well. This result can also be verified from the standard deviation of simulated data: for the log-level data, std(consumption)=0.0214, std(output)=0.0314, std(investment)=0.0919 (See Figure 2).

[11] If you specify "periods" with proper integer values, Dynare will do the simulations and stored the results in declaration order in oo_.endo_simul. But I don't know why the size of the simulation variables are larger than the periods specified in the periods option.

2.3.4 Impulse Response

All results about impulse responses are stored in $oo\_.irfs$ which contains four row vectors of size 1*40 (for level mod file). Using the definition of impulse response, we will calculate it for c_t, z_t which will again comparing to what Dynare tell us. We assume every endogenous variables stay at steady states before shock comes, i.e. at time zero.

First, we need define the impulse,i.e. the shock e_t. The technology shock e_1 is set to one standard deviation at 1st period and zero afterward.

$$e_t = \begin{cases} 0.007, & t = 1 \\ 0, & t \geqq 2 \end{cases} \tag{11}$$

Then, for the response of z_t to be calculated, Eq.(9) should be employed. The technology has steady state value zero, hence the response could be easily calculated manually. For $t = 1$, $z_t = 0.0070 * e_1 = 0.0070$ by setting $z_0 = 0$, and for $t \geq 2$, $z_t = \rho * z_{t-1} + e_t = \rho * z_{t-1}$.

At last, for the response of c_t to be calculated, Eq.(10) should be used after a little transformation. That is, the consumption c_t must be demeaned before used. Hence the constant term are dropped and we treat c_t as if it is absolute deviation from non-stochastic steady state values. For $t \geq 2$,

$$c_t = 0.027972 * k_{t-1} + 0.622876 * z_{t-1} + 0.655659 * e_t = 0.027972 * k_{t-1} + 0.622876 * z_{t-1}.$$

and $c_1 = 0.655659 * e_1 = 0.004590$.

The following is the Matlab code that reproduces the responses which is the same as Dynare produced, see Fig.3 for results.

```
1 //first for technology z_t
2 ee=zeros(1,40); //impulse
3 ee(1,1)=s; //impulse in the first period
4 ze=zeros(1,40);
5 ze(1,1)=rho*0+ee(1,1);
6 for t=2:40
7         ze(1,t)=rho*ze(1,t-1)+ee(1,t);
8 end
9 [ze',oo_.irfs.z_e']
```

```
10
11 //second for consumption c_t
12 cc=zeros(1,40);
13 cc(1,1)=0.004590;
14 kkzz=[oo_.irfs.k_e;oo_.irfs.z_e]
15 for t=2:40
16         cc(1,t)=[0.027972 0.622876]*kkzz(:,t-1);
17 end
18 [cc;oo_.irfs.c_e]'    //the two columns should be the same
```

However, there is a more convenient way to replicate the impulse responses of all endogenous variables and this is actually what Dynare does in calculating the impulse responses. In fact, using the state space representation mentioned above, the model can be written explicitly as follows:

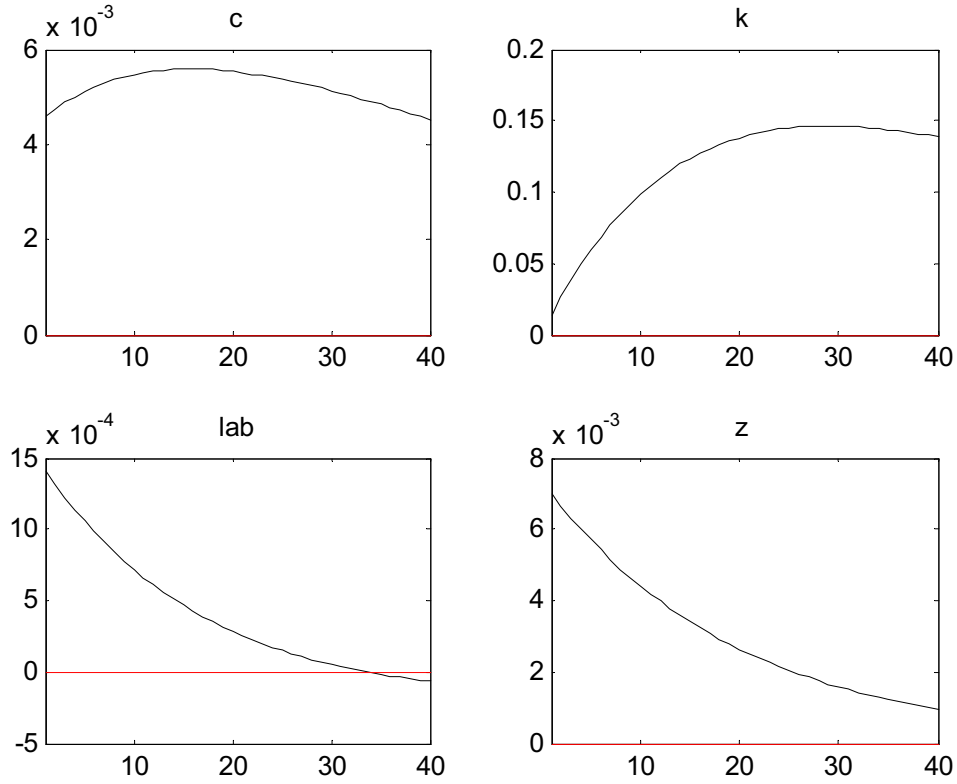
$$\begin{pmatrix} k_t \\ z_t \\ c_t \\ lab_t \end{pmatrix} = E \begin{pmatrix} k_{t-1} \\ z_{t-1} \end{pmatrix} + Fe_t \tag{12}$$

where matrix $E = oo\_.dr.ghx, F = oo\_.dr.ghu$. And the endogenous variables in Eq.(12) denote that absolute deviations (i.e. level minus its steady state value, if absolute deviation divided by steady state values, it becomes percentage deviation) , hence, at time zero, all endogenous variables are zeros since we assume all endogenous variables at steady states. Using Eq.(12), it is easy to compute the time one values of the endogenous variables. And then you can recursively compute the responses for all variables. Here is how we do in Matlab (see Fig.4,5 for results):

```
%% impulse response for level variable mod file
%please load the dynare estimation results of level variable mod file:
%GrowthApproximate_results.mat first before running this cell
load GrowthApproximate_results.mat
matE=oo_.dr.ghx;
matF=oo_.dr.ghu;
T=40;
matIRFS=zeros(size(matE,1),T);
matIRFS(:,1)=matF*s;
for ii=2:T
        matIRFS(:,ii)=matE*matIRFS(1:2,ii-1);
```

```
end
```

Figure 3: Impulse Responses of endogenous variables:Absolute Deviation by Dynare



Note: the responses in the figure (produced by Dynare itself) are absolute deviations from their corresponding steady state values not the percentage deviation since we use the level variables in mod file. Hence, there is no meaning to comparing the irfs of the endogenous variables of the absolute deviations and it is only meaningful to compare the irfs in percentage deviation. If you want to get the percentage deviations, you should divide the absolute deviation by their steady state values. If the log-level variables in the mod file, then the figure produced by Dynare are the responses of percentage deviations.

2.3.5 Various Descriptive Statistics

Moments are computed by dropping the first 100 samples which is the default value to be dropped. You can set the $options\_.drop = 0$ in *stoch_simul* command without dropping any samples. You can manually calculate this moments by dropping the first 100 samples to get the same results reported by Dynare. The following are the one reported by Dynare.

```
MOMENTS OF SIMULATED VARIABLES
VARIABLE        MEAN          STD. DEV.      VARIANCE       SKEWNESS       KURTOSIS
c               1.472131      0.029172       0.000851       -0.169043      -0.365976
k              28.754572      0.749351       0.561526       -0.192259      -0.313680
lab             0.291148      0.004043       0.000016        0.145342      -0.391494
z              -0.007047      0.021499       0.000462        0.010331      -0.610077

CORRELATION OF SIMULATED VARIABLES
VARIABLE        c         k         lab       z
c               1.0000    0.9149    0.4914    0.7419
k               0.9149    1.0000    0.0980    0.4081
lab             0.4914    0.0980    1.0000    0.9485
z               0.7419    0.4081    0.9485    1.0000
AUTOCORRELATION OF SIMULATED VARIABLES
VARIABLE        1         2         3         4         5
c               0.9874    0.9741    0.9601    0.9445    0.9270
k               0.9975    0.9936    0.9872    0.9785    0.9676
lab             0.9421    0.8868    0.8389    0.7925    0.7456
z               0.9502    0.9023    0.8604    0.8194    0.7776
Total computing time : 0h00m17s
```

2.4 Writing Your Own Steady State File

To be updated!

2.5 Comparison of Steady State Values and IRF of Level and Log-Level Variables

In this section, we will show some estimation results of the two mod files, i.e. in level and log-level variable. Table 1 shows the steady state values of level and log level variables. It is clear that after taking $exp(.)$ of log-level steady state values, the results are the same.

As we argue before, It is more convenient to use log-level variable since the IRF produced by Dyanre are percentage deviation. This point can be seen from Eq.(2)[12] and further be verified in Listing 3. However, this by no means say that use level variable in mod file is not a choice. At least, level variables in mod file is much more clear and natural in declaration and easy to read. Only a piece of codes is needed after estimation to produce the the same percentage deviation of IRF as you can see in Listing 3.

So to summarize, no matter what kind of variables(level, log-level or log-linear), it will make no differences. That is to say, in essence, the mod declarations are the same. It is your choice of model declaration. Only a slight effort involved after estimation to produce the same results.

[12] If the variable are in log-levels, then $y_t - y^s$ is the log difference, i.e. the percentage deviation.

Table 1: Steady State Values of Level and Log-level variables

variable	level	log-level	$exp(log-level)$
y	-	0.6114	1.8431
c	1.4916	0.3999	1.4916
k	29.2885	3.3772	29.2890
i	-	-1.0456	0.3515
lab	0.2916	-1.2324	0.2916
z	0	0	1.0000

Listing 3: IRF for consumption

```
%level results
load GrowthApproximate_results.mat
ys1=oo_.dr.ys;
varname1=M_.endo_names;
irf1=100*oo_.irfs.c_e/ys1(1);

%log-level results
load GrowthApproximate_exp_results.mat
ys2=oo_.dr.ys ys2exp=exp(oo_.dr.ys);
varname2=M_.endo_names;
irf2=100*oo_.irfs.c_e;
[irf1' irf2']

//Percentage deviation for both level and log-level variables
//They are the same.
0.3077     0.3077
0.3179     0.3179
0.3271     0.3271
0.3352     0.3352
0.3424     0.3424
0.3488     0.3488
0.3543     0.3543
0.3590     0.3590
0.3630     0.3630
0.3664     0.3664
0.3691     0.3690
0.3712     0.3712
0.3727     0.3727
0.3737     0.3737
```

```
0.3743      0.3743
0.3744      0.3744
0.3741      0.3741
0.3734      0.3734
0.3724      0.3724
0.3710      0.3710
0.3693      0.3693
0.3673      0.3673
0.3651      0.3651
0.3627      0.3627
0.3600      0.3600
0.3571      0.3571
0.3540      0.3540
0.3508      0.3508
0.3474      0.3474
0.3439      0.3439
0.3402      0.3402
0.3364      0.3364......
```

2.6 2nd Simulation

However, we can conduct a 2nd order approximation. When you set option $order = 2$ in *stoch_simul* command, the following 2nd order transition and policy function with cross-terms and square-terms is obtained. Comparing to what we get in 1st order approximation, the coefficients for the state variables is the same. However, the calculation of simulated endogenous variables and impulse responses becomes much more complex than 1st order. We ignore it.

POLICY AND TRANSITION FUNCTIONS

	c	k	lab	z
Constant	1.491624	29.288524	0.291594	0
(correction)	−0.000002	0.000004	0	0
k(−1)	0.027972	0.978257	−0.001831	0
z(−1)	0.622876	1.853488	0.191284	0.950000
e	0.004590	0.013657	0.001409	0.007000
k(−1),k(−1)	−0.000182	−0.000086	0.000025	0
z(−1),k(−1)	0.007307	0.024441	0.000712	0
z(−1),z(−1)	0.237404	1.157808	−0.008033	0
e,e	0.000013	0.000063	0	0
k(−1),e	0.000054	0.000180	0.000005	0
z(−1),e	0.003499	0.017062	−0.000118	0

3 Deterministic Simulation

simul triggers the computation of a deterministic simulation of the model for the number of periods set in the option periods.

Here we copy the model used in RTM v2, chap 11 by Ljungqvist and Sargent(2004). It is a deterministic growth model has inelastic labor supply, an exogenous stream of government expenditures, and several kinds of distorting taxes. A representative agent maximizes

$$\sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\gamma}}{1-\gamma}$$

Feasible allocation satisfy

$$g_t + c_t + k_{t+1} \leq A k_t^{\alpha} + (1-\delta) k_t.$$

The household budget constraint is

$$\sum_{t=0}^{\infty} (q_t(1+\tau_{ct})c_t + (1-\tau_{it})q_t(k_{t+1} - (1-\delta)k_t)) \leq \sum_{t=0}^{\infty} (r_t(1-\tau_{kt})k_t + w_t).$$

and the government budget constraint is

$$\sum_{t=0}^{\infty} q_t g_t \leq \sum_{t=0}^{\infty} (\tau_{ct} q_t c_t - \tau_{it} q_t (k_{t+1} - (1-\delta)k_t) + r_t(1-\tau_{kt})k_t + w_t).$$

The following conditions characterize an equilibrium:

$$\begin{aligned}
c_t &= A k_t^{\alpha} + (1-\delta)k_t - k_{t+1} - g_t \\
q_t &= \beta^t c_t^{-\gamma}/(1+\tau_{ct}) \\
r_t/q_t &= A\alpha k_t^{\alpha-1} \\
w_t/q_t &= A k_t^{\alpha} - k_t A \alpha k_t^{\alpha-1} \\
R_{t+1} &= \frac{1+\tau_{ct}}{1+\tau_{ct+1}} \left(\frac{1+\tau_{it+1}}{1+\tau_{it}}(1-\delta) + \frac{1-\tau_{kt}}{1-\tau_{it}} A\alpha k_{t+1}^{\alpha-1} \right) \\
s_t/q_t &= (1-\tau_{kt}) A \alpha k_t^{\alpha-1} + 1 - \delta \\
c_t^{-\gamma} &= \beta c_{t+1}^{-\gamma} R_{t+1}
\end{aligned}$$

```
1  var c k;
2  varexo taui tauc tauk g;
3  parameters bet gam del alpha A;
4  bet=.95;
5  gam=2;
6  del=.2;
7  alpha=.33;
8  A=1;
9
10 model;
11 k=A*k(-1)^alpha+(1-del)*k(-1)-c-g;
12 c^(-gam)=bet*(c(+1)^(-gam))*((1+tauc(-1))/(1+tauc))*((1-taui)*(1-
       del)/(1-taui(-1))+   ((1-tauk)/(1-taui(-1)))*alpha*A*k(-1)^(alpha
       -1));
13 end;
```

```
14
15 initval;
16 k=1.5;
17 c=0.6;
18 g = 0.2;
19 tauc = 0;
20 taui = 0;
21 tauk = 0;
22 end;
23 steady;
24
25 endval;
26 k=1.5;
27 c=0.4;
28 g =.4;
29 tauc =0;
30 taui =0;
31 tauk =0;
32 end;
33 steady;
34
35 shocks;
36 var g;
37 periods 1:9;
38 values 0.2;
39 end;
40
41 simul(periods=100);
42
43 //post estimation analysis
44 co=ys0_(1,1);
45 ko = ys0_(2,1);
46 go = ex0_(4,1);
47 rbig0=1/bet;
48 rbig=oo_.endo_simul(1,2:101).^(-gam)./(bet*oo_.endo_simul(1,3:102)
       .^(-gam)); rq0=alpha*A*ko^(alpha-1);
49 rq=alpha*A*oo_.endo_simul(2,1:100).^(alpha-1);
50 wq0=A*ko^alpha-ko*alpha*A*ko^(alpha-1);
51 wq=A*oo_.endo_simul(2,1:100).^alpha-oo_.endo_simul(2,1:100).*alpha*
       A.*oo_.endo_simul(2,1:100).^(alpha-1);
52 sq0=(1-oo_.exo_simul(1,3))*A*alpha*ko^(alpha-1)+(1-del);
53 sq=(1-oo_.exo_simul(1:100,3)')*A*alpha.*oo_.endo_simul(2,1:100).^(
       alpha-1)+(1-del);
54
55 figure
56 subplot(2,3,1)
```

```
57 plot([ko*ones(100,1)  oo_.endo_simul(2,1:100)' ])
58 title('k')
59 subplot(2,3,2)
60 plot([co*ones(100,1)  oo_.endo_simul(1,2:101)' ])
61 title('c')
62 subplot(2,3,3)
63 plot([rbig0*ones(100,1) rbig' ])
64 title('R')
65 subplot(2,3,4)
66 plot([wq0*ones(100,1) wq' ])
67 title('w/q')
68 subplot(2,3,5)
69 plot([sq0*ones(100,1) sq' ])
70 title('s/q')
71 subplot(2,3,6)
72 plot([rq0*ones(100,1) rq' ])
73 title('r/q')
```

4 Estimation

Provided that you have observations on some endogenous variables, it is possible to use Dynare to estimate some or all parameters. Both maximum likelihood and Bayesian techniques are available.

Note that in order to avoid stochastic singularity, you must have at least as many shocks or measurement errors in your model as you have observed variables.

References

Barillas, Francisco, Anmol Bhandari, Riccardo Colacito, Sagiri Kitao, Christian Matthes, Thomas J. Sarget and Yongseok Shin. 2010. "Practicing Dynare." *New York University* .

Dynare_Team. 2013. "Dynare Reference Manual,v4.3.3." *http://www.Dynare.org* .

Sims, Eric. 2012. "Graduate Macro Theory II: Notes on Using Dynare." *University of Notre Dame* .

Figure 2: Simulation for various endogenous variables: in deviation and log-level

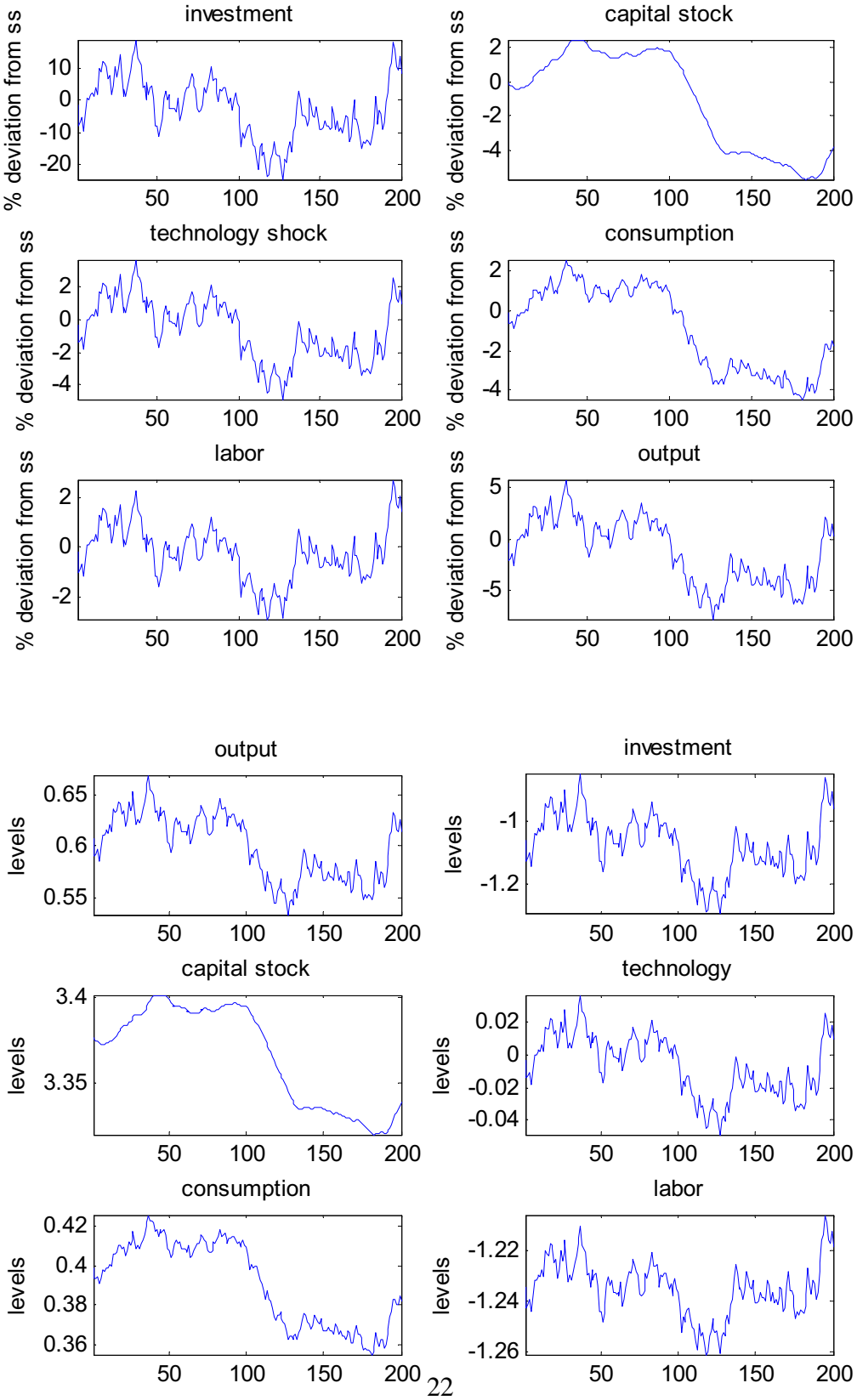


Figure 4: Impulse Responses of endogenous variables:Absolute and Percentage Deviation by Calculation

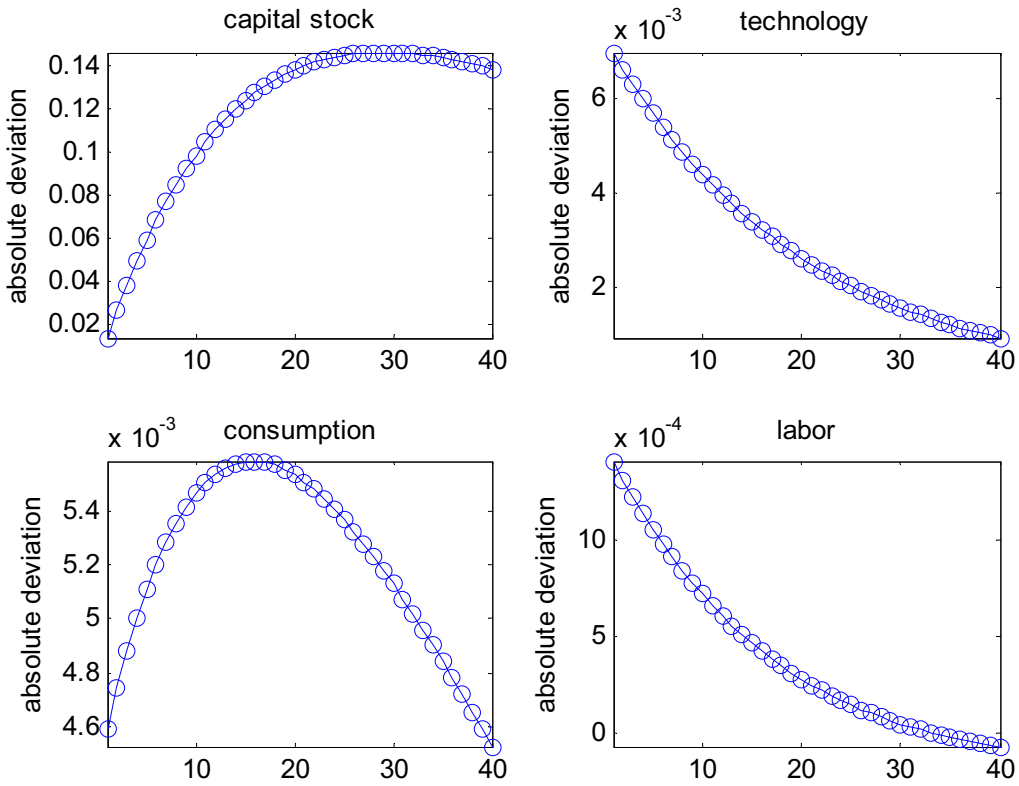


Figure 5: Impulse Responses of endogenous variables:Percentage Deviation by Calculation