Graduate Macro Theory II:
Notes on Using Dynare

Eric Sims
University of Notre Dame

Spring 2012

1 Introduction

This document will present some simple examples of how to solve and simulate, DSGE models using Dynare. Dynare can also be used to estimate the parameters of DSGE models via Maximum Likelihood or Bayesian Maximum Likelihood, but that is beyond the purview of a first year course.

Dynare is not its own program but is rather basically a collection of Matlab codes. To run Dynare, you must first install it. You can download it from the following website: `http://www.dynare.org/download`. I will only support downloads using Matlab. You need to choose a destination folder. My destination folder is in the C drive and is called "dynare". I have version 4.1.1. The full path is: "C:\dynare \4.1.1 \matlab ". There are newer versions of Dynare available and you should download the latest stable version.

To run Dynare, you have to create .mod files (you can create these in the m-file editor, but be sure to save them as .mod, not as .m files). Then, to run your code, simply type "dynare filename" into the command window (or you can have this command within a seperate m-file which can be in another directory so long as you call up the Dynare directory). Equivalently, you can "set the path" in Matlab so that you don't always have to call up the Dynare directory.

2 A Neoclassical Model with Fixed Labor

Consider a simple stochastic neoclassical model with fixed labor as the laboratory. The planner's problem can be written:

$$\max \quad E_0 \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\sigma} - 1}{1-\sigma}$$

s.t.

$$k_{t+1} = a_t k_t^{\alpha} - c_t + (1-\delta)k_t$$

TFP is assumed to follow a mean zero AR(1) in the log:

$$\ln a_t = \rho \ln a_{t-1} + \varepsilon_t$$

The first order conditions for this problem can be characterized with three non-linear difference equations and a transversality condition:

$$\begin{aligned} c_t^{-\sigma} &= \beta E_t c_{t+1}^{-\sigma}(\alpha a_{t+1} k_{t+1}^{\alpha-1} + (1-\delta)) \\ k_{t+1} &= a_t k_t^{\alpha} - c_t + (1-\delta)k_t \\ \ln a_t &= \rho \ln a_{t-1} + \varepsilon_t \\ \lim_{t\to\infty} \beta^t c_t^{-\sigma} k_{t+1} &= 0 \end{aligned}$$

In addition, I have two auxiliary "static" variables that define output from the production function and investment from the accounting identity:

$$\begin{aligned} y_t &= a_t k_t^{\alpha} \\ i_t &= y_t - c_t \end{aligned}$$

The parameters that need to be calibrated are σ, α, δ, β, ρ, and σ_ε. I set the coefficient of relative risk aversion to 1 (log), capital's share to 1/3, the depreciation rate to 0.025, the discount factor to 0.99, the persistence of technology shocks to 0.95, and the standard deviation of technology shocks to 0.01 (i.e. one percent).

2.1 A Timing Convention

There is a timing convention in Dynare that requires that we (slightly) rewrite the model. In particular, Dynare requires that predetermined variables (like the capital stock) show up as dated $t-1$ in the time t equations and t in the $t+1$ equations. This is how you "tell" Dynare if a variable is a state variable or not. As such, we need to rewrite the FOCs:

$$\begin{aligned} c_t^{-\sigma} &= \beta E_t c_{t+1}^{-\sigma}(\alpha a_{t+1} k_t^{\alpha-1} + (1-\delta)) \\ k_t &= a_t k_{t-1}^{\alpha} - c_t + (1-\delta)k_{t-1} \\ \ln a_t &= \rho \ln a_{t-1} + \varepsilon_t \\ \lim_{t\to\infty} \beta^t c_t^{-\sigma} k_t &= 0 \\ y_t &= a_t k_{t-1}^{\alpha} \\ i_t &= y_t - c_t \end{aligned}$$

In other words, we essentially just lag the capital stock one period in each relevant equation. k_t is just re-interpreted as k_{t+1}.

3 Writing the Dynare Code

With the first order conditions in hand, we are now ready to begin to write our dynare code (which occurs in the .mod file). The dynare code will not run unless each entry is followed by a semicolon to suppress the output.

The first step is to declare the variables of the model. Dynare automatically sorts variables into the following order: static variables (i.e. only enter FOCs at time t), purely predetermined variables (only enter FOCs at t and $t-1$), variables that are both predetermined and forward-looking (appear at $t-1$, t, and $t+1$), and variables that are purely forward-looking (only appear at dates t and $t+1$). In our model, output is static, capital is purely predetermined, technology is both predetermined and forward-looking (since future TFP shows up in the Euler equation), and consumption is forward-looking. It makes sense to declare the variables in this order. The first non-comment command in your Dynare code should be: "var" followed by the endogenous variable names and culminating with a semicolon. For the purposes of naming things, everything other than the shock(s) is considered to be endogenous My first line of code looks like:

```
1 var y i k a c;
```

The next line of code specifies the exogenous variables. These are just the shocks, and in this model I only have one:

```
1 varexo e;
```

The next step is to declare the parameters of the model. You simply type "parameters" followed by parameter names:

```
1 parameters alpha beta ∆ rho sigma sigmae;
```

Immediately following this command, you need to specify numerical values for these parameters:

```
1 alpha = 0.33;
2 beta = 0.99;
3 ∆ = 0.025;
4 rho = 0.95;
5 sigma = 1;
6 sigmae = 0.01;
```

Now that you've named variables and specified parameter values, it's now time to "declare the model". To do this, you type in "model;" followed by the first order conditions, constraints, identities, etc., followed by "end;". We typically want to approximate the solutions of the natural logs of the variables, so that impulse responses, etc. are all in percentage terms. Dynare will do linear approximations 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 shows up in most of the FOCs. Then you just type in the first order conditions. If a variable appears dated t, then you just type x. If a variable is $t+1$, you type $x(+1)$; if it's $t-1$, then $x(-1)$. If, for whatever reason, you need variables beyond one period away to appear, you could define auxiliary state variables to make the system work out this way. My first order conditions for the above model look like:

```
model;

exp(c)^(-sigma) = beta*(exp(c(+1))^(-sigma)*(alpha*exp(a(+1))*exp(k)^(alpha -1) + (1-Δ)));
exp(y) = exp(a)*exp(k(-1))^(alpha)*exp(n)^(1-alpha);
exp(k) = exp(i) + (1-Δ)*exp(k(-1));
exp(y) = exp(c) + exp(i);
a = rho*a(-1)+ e;

end;
```

Dynare interprets all of the underlying variables here as logs. Since I specified the stochastic process for a_t in the log, I do not need to write it using the exponential. Dynare will solve for the steady state numerically for you. To get it to do this you need to give it guess of the steady state. If you give it bad guesses, Dynare may not converge and you can get an error. When giving initial values, remember that Dynare is interpreting all the variables as logs. Hence, if you know that steady state capital is 30, you want to give an initial value of capital somewhere near $\ln 30$, not 30. you begin this part of the code with the command "initval;", followed by the initial values/guesses of all the endogenous variables, capped off with "end". My code for this part looks like:

```
initval;

k = log(29);
y = log(3);
a = 0;
c = log(2.5);
i = log(1.5);

end;
```

The next step is to specify the variance of the shocks. This part of the code starts with "shocks;",

followed by a specification of the *variance* (not standard deviation), followed by "end;":

```
var e = sigmae^2;
```

In the next step you simply type in "steady;". This command calculates the steady state values of the endogenous variables of the model:

```
steady;
```

The next command is the payoff. It's the "stoch_simul" command, which is what solves the model, produces the policy functions, and generates impulse responses functions and unconditional second moments. There are a number of options following this command. If you just type in "stoch_simul;", it's going to give you the default output. The default output is: (1) steady state values of endogenous variables; (2) a model summary, which counts variables by type; (3) covariance matrix of shocks (which in the example I'm giving is a scalar); (4) the policy and transition functions (in state space notation); (5) theoretical first and second moments; (6) a theoretical correlation matrix; (7) theoretical autocovariances up to order 5; and (8) impulse responses. By default, Dynare does a second order approximation about the steady state; the second order approximation involves "cross terms" , and so the policy/transition functions look more complicated than you're used to seeing.

There are a number of options one can include behind "stoch_simul;" (to do this you type "stoch_simul(options);". There are several of these that you can read about in the manual, but the more important ones for our purposes are:

- hp_filter = integer: this will produce theoretical moments (variances, covariances, autocorrelations) after HP filtering the data (the default is to apply no filter to the data). We typically want to look at HP filtered moments, so this is an option you'll want to use. The integer is a number (i.e. 1600) corresponding to the penalty parameter in the HP filter. So, for example, typing "stoch_simul(hp_filter=1600);" will produce theoretical (i.e. analytical) moments of the HP filtered data. The only problem here is that it will not simultaneously do HP filtering of simulated data; you can either get moments from simulated data or analytical HP filtered moments

- irf = integer: this will change the number of periods plotted in the impulse response functions. The default value is 40. To suppress printing impulse responses altogether, type in 0 for the number of horizons. For example, to see impulse responses for only 20 periods, type "stoch_simul(irf=20);".

- nocorr: this will mean it will not print a matrix of correlations in the command window

- nofunctions: this will suppress printing the state space coefficients of the solution in the command window

- nomoments: this will suppress printing of moments
- noprint: this will suppress printing of any output at all
- order = 1 or 2: this tells Dynare the order of the (log) approximation. The default is a second order approximation. Hence, typing "order=1" will have it do a linear approximation.
- periods = integer: Dynare's default is to produce analytical/theoretical moments of the variables. Having periods not equal to zero will instead have it simulate data and take the moments from the simulated data. By default, Dynare drops the first 100 values from a simulation, so you need to give it a number of periods greater than 100 for this to work. Hence, typing "stoch_simul(periods=300);" will produce moments based on a simulation with 200 variables.
- drop = integer: You can change the number of observations to drop from the simulations. For example, typing "stoch_simul(drop=0);" will result in no observations being dropped in the simulations.
- simul_seed = integer: sets the seed used in the random number generator for the simulations.

For example, typing "stoch_simul(nofunctions,hp_filter=1600,order=1,irf=20);" will suppress the policy function output, will produce analytical HP filtered moments, will do a first order approximation, and will plot 20 periods in the impulse responses.

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 (here $m = 2$) and x_t be a $n \times 1$ vector of controls (here I include "static" controls like output as well as dynamic controls; you don't need to eliminate the static controls to use Dynare), which in this model is $m = 3$. The state space representation of the system can be written:

$$s_t = As_{t-1} + B\varepsilon_t$$
$$x_t = \Phi s_t$$

Above A is $m \times m$, and can be found as discussed previously in class. ε_t is $w \times 1$; so in this model $w = 1$; and B is $m \times w$. Φ is the policy function, and is $n \times m$. Dynare writes the system out by substituting the state transition equation into the policy function as follows:

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

It is easily verified that these matrix multiplications are permissable; Φ is $n \times m$ and A is $m \times m$. Hence ΦA, call it C, is $n \times m$. B is $m \times w$; hence ΦB is $n \times w$. Call $\Phi B = D$. Then we can write out the full system as:

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

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

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

3.1 The Full .mod File

Below is my code in the .mod file in its entirety. Note that you can comment in a .mod file just like you can in a regular .m file.

```
% Basic Dynare code for grad macro

% Eric Sims
% University of Notre Dame
% Spring 2011

var y i k a c;
varexo e;

parameters alpha beta Δ rho sigma sigmae;

alpha = 0.33;
beta = 0.99;
Δ = 0.025;
rho = 0.95;
sigma = 1;
sigmae = 0.01;

model;

exp(c)^(-sigma) = beta*(exp(c(+1))^(-sigma)*(alpha*exp(a(+1))*exp(k)^(alpha -1) + (1-Δ)));
exp(y) = exp(a)*exp(k(-1))^(alpha);
exp(k) = exp(i) + (1-Δ)*exp(k(-1));
exp(y) = exp(c) + exp(i);
a = rho*a(-1)+ e;

end;

initval;

```

```
k = log(29);
y = log(3);
a = 0;
c = log(2.5);
i = log(1.5);

end;

shocks;
var e = sigmae^2;
end;

steady;

stoch_simul(hp_filter=1600,order=1,irf=40);
```

4 Running Dynare

To run Dynare you can't hit F5 or type the name of the .mod file into the command prompt. You have to type "dynare filename" (without the .mod on the end of the file name). For this to run you must either be in the same directory where your Dynare files are stored, you must have set the path of where your Dynare files are stored, or you must type "addpath (directory)" where the directory is the location of your Dynare files. It is also helpful to add "noclearall" to the declaration so that you are typing in "dynare filename noclearall". Otherwise Dynare will automatically erase all previous output, which you may not want.

So as to do other things, I usually write a .m file to go along with my .mod file. My code for the .m file is simple:

```
% Running Dynare

addpath c:\dynare\4.1.1\matlab

dynare basic_nc
```

5 Dynare Output

Dynare will produce a bunch of output in the command window for you (unless you include options discussed above to suppress it). Below is the output that I get:

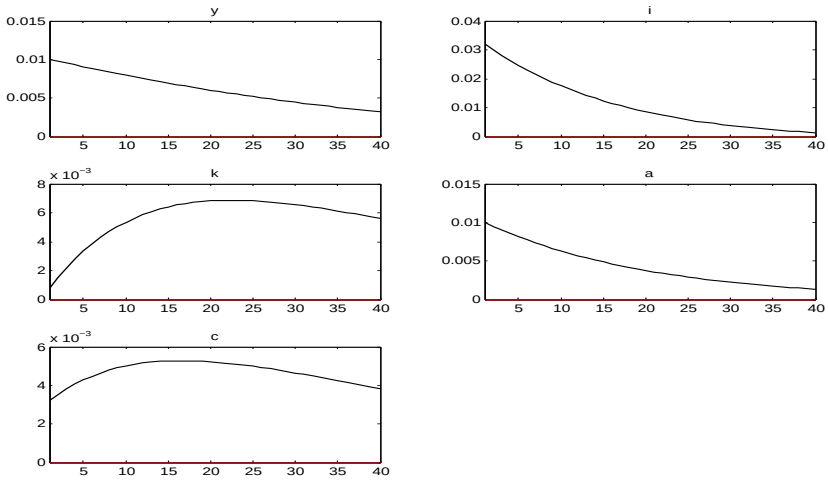
```
Configuring Dynare ...
[mex] Generalized QZ.
[mex] Sylvester equation solution.
```

```
[mex] Kronecker products.
[mex] Sparse kronecker products.
[mex] Bytecode evaluation.
[mex] k-order perturbation solver.
[mex] k-order solution simulation.

Starting Dynare (version 4.1.1).
Starting preprocessing of the model file ...
Found 5 equation(s).
Evaluating expressions...done
Computing static model derivatives:
 - order 1
Computing dynamic model derivatives:
 - order 1
Processing outputs ...done
Preprocessing completed.
Starting MATLAB/Octave computing.


STEADY-STATE RESULTS:

y        1.10371
i        -0.344308
k        3.34457
a        0
c        0.835782

MODEL SUMMARY

  Number of variables:         5
  Number of stochastic shocks: 1
  Number of state variables:   2
  Number of jumpers:           2
  Number of static variables:  2


MATRIX OF COVARIANCE OF EXOGENOUS SHOCKS

Variables    e
e            0.000100

POLICY AND TRANSITION FUNCTIONS
                  y           i               k           a           c
Constant     1.103709    -0.344308      3.344571           0     0.835782
k(-1)        0.330000    -0.517541      0.962061           0     0.590408
a(-1)        0.950000     3.043702      0.076093    0.950000     0.306708
e            1.000000     3.203897      0.080097    1.000000     0.322850


```

```
THEORETICAL MOMENTS (HP filter, lambda = 1600)

VARIABLE    MEAN       STD. DEV.  VARIANCE
y              1.1037     0.0131     0.0002
i             -0.3443     0.0418     0.0017
k              3.3446     0.0037     0.0000
a              0.0000     0.0130     0.0002
c              0.8358     0.0047     0.0000



MATRIX OF CORRELATIONS (HP filter, lambda = 1600)

Variables    y        i        k        a        c
y            1.0000  0.9903  0.3618  0.9956  0.9256
i            0.9903  1.0000  0.2285  0.9989  0.8640
k            0.3618  0.2285  1.0000  0.2730  0.6877
a            0.9956  0.9989  0.2730  1.0000  0.8862
c            0.9256  0.8640  0.6877  0.8862  1.0000



COEFFICIENTS OF AUTOCORRELATION (HP filter, lambda = 1600)

Order    1       2       3       4       5
y        0.7217  0.4846  0.2868  0.1257 -0.0018
i        0.7107  0.4670  0.2663  0.1050 -0.0208
k        0.9596  0.8619  0.7268  0.5704  0.4058
a        0.7133  0.4711  0.2711  0.1098 -0.0163
c        0.7941  0.5994  0.4206  0.2610  0.1222
Total computing time : 0h00m10s
```

The "constant" in the policy and transition functions output is just the steady state values; the coefficients of the (modified) state space representation are easy to understand. In addition, Dynare produces means (steady state values), volatilities (standard deviations), and variances for each endogenous variable. It also produces the matrix of correlations as well as coefficients of autocorrelation at different lags. Given the options I specified, all of these moments are (i) theoretical (in the sense of being analytical, not based on a simulation) and (ii) based on HP filtered data.

Dynare also produces impulse responses to the shocks. These are shown here:



There is one interpretive issue in the IRFs of which you need to be aware. Recall that Dynare interprets k_t as k_{t+1}. Hence, the impulse response of k_t that it produces is really the impulse response of k_{t+1}. To get the impulse responses of k_t, you need to (i) add a zero as the impact response (remember the capital stock can't react on impact as it is predetermined) and (ii) set the response of k_t at horizon h equal to the response of what Dynare gives as the impulse response in period $h+1$.

6 Where is the Output Stored?

Dynare is great at producing lots of different output right there for you to see. It's a little more challenging to find where your output is stored.

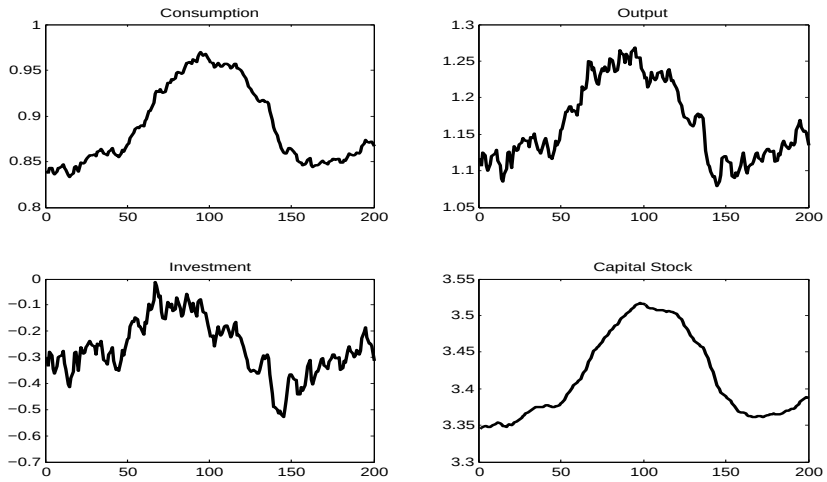
Dynare will store IRFs for you under the following nomenclature: "x_e" denotes the impulse response of variable x to shock e. The coefficients of the state space representation are a bit harder to find. In terms of the notation introduced above, the coefficients of the Ψ matrix are stored in "oo_.dr.ghx"; the coefficients of the Ω matrix are stored in "oo_.dr.ghu"; and the steady state values are stored in "oo_dr.ys".

It is helpful to know where these are in the event that you want to conduct your own simulation. I can simulate out variables just as before by drawing shocks from some distribution. For the simulation I assume that everything begins in steady state, so that the initial conditions are all 0 (recall that the variables are deviations about steady state). Here is some code to do that:

```
m = 2; % number of states
n = 3; % number of controls

psi = oo_.dr.ghx;
omega = oo_.dr.ghu;
ss = oo_.dr.ys;
```

```
7
8  T = 200; % number of periods to simulate
9
10 es = sigmae*randn(T,1);
11 Xsim = zeros(n+m,T);
12
13 Xsim(:,1) = omega*es(1,1);
14
15 for j = 2:T
16     Xsim(:,j) = psi*Xsim(3:4,j-1) + omega*es(j,1);
17 end
18
19 % add back in steady state values so that these are expressed as log
20 % levels, not log deviatins
21 for j = 1:T
22 Xsim(:,j) = Xsim(:,j) + ss;
23 end
```

Here is a figure of the simulated time paths of these variables:



The simulations 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.

Dynare will also directly do simulations for you if you want. For example, if I type in "stoch_simul(order=1,irf=40,periods=200)" it will generate a simulation of 200 periods of the variables of the model (actually it will produce a simulation of 203 periods; I don't fully understand why it adds the three periods). If you use this command then it produces moments (standard deviations, correlations, and autocorrelations) off of the simulated data.

7 Advanced Topics

In this section I discuss a couple of more "advanced" issues when using Dynare.

7.1 From .mod file to .m file

As you'll see Dynare runs this kind of annoying start up script when you execute a .mod file. After running a .mod file once Dynare produces a .m file with the same name as the .mod file. For example, if my .mod file is called "in_class_dynare.mod" it will produce a .m file called "in_class_dynare.m".

Once you've run the .mod file once you can then just run the .m file on subsequent attempts. This will be a bit faster. You can actually edit the .m file but you want to be careful and be sure that you know what you are doing. For example, sometimes I get a crash where Matlab reports an error that it can't open a log file. I can just go into the .m file and comment out the lines corresponding to the log file creation. Also, you can comment out the "display amount of time" command at the end.

7.2 Specifying Parameters Outside the .mod File

Sometimes you will want to loop over different values of parameters or otherwise specify them somewhere else. Fortunately, this is pretty easy to do in Dynare.

Suppose you start in a .m file and want to specify the parameters of the model there. I would write the following code:

```
alpha = 0.33;
beta = 0.99;
Δ = 0.025;
rho = 0.95;
sigma = 1;
sigmae = 1;

save parameterfile alpha beta Δ rho sigma sigmae
```

The last command saves the parameters as .mat file. You can modify the .mod file to accept these parameters as follows:

```
parameters alpha beta Δ rho sigma sigmae;

load parameterfile;
set_param_value('alpha',alpha)
set_param_value('beta',beta)
set_param_value('Δ',Δ)
set_param_value('rho',rho)
set_param_value('sigma',sigma)
set_param_value('sigmae',sigmae)
```

The .mod file should run using the parameters saved in parameterfile.mat. You can, of course, override those parameters within the structure of the .mod file. What is nice about doing this is that, once you've run the .mod file once for a given set of parameters, if your code is specified this way you can run it again for different parameters by just running the generated .m file and it will use the parameter values. If you aren't saving the parameters values, if you run the .mod file once, then change the parameters in the .mod file and then run the .m file, the .m file will use the original parameters from when the .mod file was first run. This way allows you to get around that. For example, you can use this to write loops. You can solve the model for a bunch of different parameters values and save some statistics from each run, for example.

7.3 Writing Your Own Steady State File

By default Dynare uses a numerical algorithm to compute the steady state of your model. This can take time but can also fail if you give it bad initial values. This can become particularly problematic if you are looping over many different parameter values which change the steady state, which can mean that the initial conditions which are "good" for one parameter configuration are not good for another, and can cause the Dynare file to not work.

For most models it is not difficult to solve for the steady state by hand; for the growth model it is particularly straightforward. Fortunately Dynare allows you to write a separate .m file that can compute the steady state given parameter files. It needs to have the same name as the .mod file followed by "_steadystate". So, for example, if my .mod file is called "filename.mod" I need to name the steady state file "filename_steadystate.m".

The general structure of the steady state file needs to look as follows (all of this information needs to be there). You also need to eliminate (or comment out) the "initval" component of the .mod file if you are going to specify your own steady state file.

```
1  function [ys,check] = in_class_dynare_parameters_saved_steadystate(ys,exe);
2
3  global M_
4
5  alpha = M_.params(1);
6  beta = M_.params(2);
7  Δ = M_.params(3);
8  rho = M_.params(4);
9  sigma = M_.params(5);
10 sigmae = M_.params(6);
11
12 k = (alpha/(1/beta - (1-Δ)))^(1/(1-alpha));
13 y = k^(alpha);
14 i = Δ*k;
15 c = y - i;
16 a = 1;
17
```

```
18 check = 0;
19
20 ys = [log(y)
21      log(i)
22      log(k)
23      log(a)
24      log(c)];
```