An Introduction to Dynare

Yang Guang

Department of Economics
Nankai University

07.07.2022 / Kunming

Contents

Contents

What is Dynare

- Dynare is a software for handling a wide class of economics models, in particular DSGE models and OLG models
- Dynare offers a user-friendly and intuitive way of describing these models.
- Dynare is a free software, available for the Windows, macOs, and linux platforms.

Installation

Packaged versions of Dynare available for Windows (8.1, 10 and 11).

- In order to run Dynare, you need one of the following:
  - MATLAB, any version ranging from 8.3(R2014a) to 9.12(R2022a).
  - GNU Octave, any version ranging from 5.2.0 to 6.4.0, with the statistics package from Octave-Forge.
- Configuration
  - under Windows, using the addpath command in the MATLAB command window

```
addpath c:/dynare/versions/matlab
```

Invocation

- Dynare is invoked using the dynare command at the MATLAB or Octave

```
dynare FILENAME[.mod] [OPTIONS]
```

- This command launches Dynare and executes the instructions included in FILENAME.mod. This user-supplied file contains the model and the processing instructions, as described in *The model file*.

Contents

Conventions

- A model file contains a list of commands and of blocks.
  - Each command and each element of a block is terminated by a semicolon (;). Blocks are terminated by end;
  - Single-line comments begin with // and stop at the end of the line.

```
// This is a single comment
```

```
var x; // This is a comment about x
```

Variable declaration

- Dynare allows the user to choose their own variable names.
  - Commands for declaring endogenous variables are described below

```
Command: var VAR_NAME [OPTIONS];
Example: var c $c$ {long_name='Consumption
    '};
```

  - Commands for declaring exodogenous variables are described below

```
Command: varexo VAR_NAME [OPTIONS];
Example: varexo m gov;
```

Variable declaration

- Dynare allows the user to choose their own parameter names.
  - Commands for declaring parameters are described below

```
Command: parameters PARAM_NAMES [OPTIONS];
Example: parameters alpha beta;
```

- For using Dynare for computing simulations, it is necessary to calibrate the parameters of the model.
  - The syntax is the following:

```
Example: parameters alpha beta;
         beta = 0.99;
         alpha =0.36;
```

Model declaration

- The model is declared inside a model block.

```
Block:  model;
```

- it is possible to name the equations with a name-tag, using a syntax like:

```
model;
[name = 'Budget constraint'];
c + k = k^theta*A;
end;
```

Model declaration

- Inside the model block, Dynare allows the creation of model-local variables, which constitute a simple way to share a common expression between several equations.

```
model;
# gamma = 1 - 1/sigma;
u1 = c1^gamma/gamma;
u2 = c2^gamma/gamma;
end;
```

- if the model is declared as being linear.It spares oneself from having to declare initial valuesfor computing the steady state of a stationary linear model.

```
model(linear);
x = a*x(-1)+b*y(+1)+e_x;
y = d*y(-1)+e_y;
end;
```

Initial and terminal declaration

- For most simulation exercises, it is necessary to provide initial (and possibly terminal) conditions. It is also necessary to provide initial guess values for non-linear solvers.
- In a deterministic model,the block initval provides values for non-linear solvers and guess values for steady state computations

```
Block: initval
example: initval;
         c = 1.2;
         k = 12;
         x = 1;
         end;
```

- if the initval block is immediately followed by a steady command,steady command will compute the steady state of the model.

Initial and terminal declaration

- In a stochastic model,the block initval only provides guess values for steady state computations

```
Block: initval
example: initval;
         c = 1.2;
         k = 12;
         x = 1;
         end;
         steady;
```

Initial and terminal declaration

- The block endval only make sense e in a deterministic model.it provides the terminal conditions for variables

```
Block: endval
example: endval;
         c = 2;
         k = 20;
         x = 2;
         end;
         steady;
```

Initial and terminal declaration

Example(1)

- In this example, the problem is finding the optimal path for consumption and capital for the periods $t = 1$ to $T = 200$.
  - c is a forward-looking variable and the exogenous variable
  - k is a purely backward-looking (state) variable.
  - exogenous technology level x appears with a lead in the expected return of physical capital.
- initial equilibrium is computed by steady conditional on x=1, and the terminal one conditional on x=2.
  - The initval block sets the initial condition for k (since it is the only backward-looking variable).
  - The endval block sets the terminal condition for c (since it is the only forward-looking endogenous variable).

Initial and terminal declaration

```
var c k;
varexo x;
model;
c + k - aa*x*k(-1)^alph - (1-delt)*k(-1);
c^(-gam) - (1+bet)^(-1)*(aa*alph*x(+1)*k^(alph-1) + 1
    - delt)*c(+1)^(-gam);
initval;
c = 1.2;
k = 12;
x = 1;
end;
steady;
endval;
c = 2;
k = 20;
x = 2;
end;
steady;
```

Initial and terminal declaration

Example(2)

- it is not necessary to specify c and x in the initval block and k in the endval block.
  - at t=1, optimization problem is to choose c(1) and k(1)(k(1) is inherited from t=0), given x(1) x(2),c(0) x(0) play no role
  - at t=201, that choice only depends on current capital as well as future consumption c and technology x, but not on future capital k.
- In this example, there is no steady command, hence the conditions are exactly those specified in the the initval and endval blocks.
- if there is steady command. steady steady specifies that those conditions before and after the simulation range are equal to being at the steady state given the exogenous variables in the initval and endval blocks.

Initial and terminal declaration

```
var c k;
varexo x;
model;
c + k - aa*x*k(-1)^alph - (1-delt)*k(-1);
c^(-gam) - (1+bet)^(-1)*(aa*alph*x(+1)*k^(alph-1) + 1
    - delt)*c(+1)^(-gam);
end;
initval;
k = 12;
end;
endval;
c = 2;
x = 1.1;
end;
perfect_foresight_setup(periods=200);
perfect_foresight_solver;
```

Shocks on exogenous variables

- In a deterministic context, if one wants to analyze the equilibrium transition, use initval and endval blocks.
- one's purpose is to study the effect of a temporary shock after which the system goes back to the original equilibrium , use shock block.
- In a stochastic framework, the exogenous variables take random values in each period, users can specify the variability of these shocks within shocks block.

```
block: shocks
```

Shocks on exogenous variables

- For deterministic simulations, the shocks block specifies temporary changes in the value of exogenous variables. For permanent shocks, use an endval block.

```
var VARIABLE_NAME;
periods INTEGER[:INTEGER] [[,] INTEGER[:INTEGER
    ]]...;
values DOUBLE | (EXPRESSION) [[,] DOUBLE | (
    EXPRESSION) ]...;
```

- Example

```
  shocks;
  var e;
  periods 1;
  values 0.5;
```

Shocks on exogenous variables

- In stochastic context, the shocks block specifies the non zero elements of the covariance matrix of the shocks of exogenous variables.

```
var VARIABLE_NAME; stderr EXPRESSION;
```

- Example

```
var u; stderr 0.009;
```

Contents

What is steady state

- In systems theory, a system or a process is in a steady state if the variables (called state variables) which define the behavior of the system or the process are unchanging in time.
  - In continuous time, the partial derivative of f with respect to time is zero $\frac{\partial f}{\partial t} = 0$
  - In discrete time, it means that the first difference of each property is zero and remains so: $f_t - f_{t-1} = 0$

Methods of finding the steady state

- there are two ways of computing the steady state.
  - use a nonlinear Newton-type solver.
  - use your knowledge of the model, by providing Dynare with a method to compute the steady state.

Methods of finding the steady state

- This command computes the steady state of a model using a nonlinear Newton-type solver and displays it.

```
Command: steady;
```

steady uses an iterative procedure and takes as initial guess the value of the endogenous variables set in the previous initval or endval block.

- If you know how to compute the steady state for your model, you can provide a MATLAB function doing the computation instead of using steady.
  - The easiest way is to write a steady_state_model block.

```
Block: steady_state_model ;
```

  - You can write the corresponding MATLAB function by hand. If your MOD-file is called FILENAME.mod, the steady state file must be called FILENAME_steadystate.m.

Methods of finding the steady state

- This command computes the steady state of a model using a nonlinear Newton-type solver and displays it.

```
Command: steady;
```

steady uses an iterative procedure and takes as initial guess the value of the endogenous variables set in the previous initval or endval block.

Methods of finding the steady state

- Example(3)
  When the analytical solution of the model is known, this command can be used to help Dynare find the steady state in a more efficient and reliable way.

```
var m P c e W R k d n l gy_obs gp_obs y dA;
varexo e_a e_m;
parameters alp bet gam mst rho psi del;
...
// parameter calibration, (dynamic) model declaration,
     shock, calibration...
...
steady_state_model;
dA = exp(gam);
gst = 1/dA; // A temporary variable
m = mst;
```

Methods of finding the steady state

```
steady_state_model;
dA = exp(gam);
gst = 1/dA; // A temporary variable
m = mst;
// Three other temporary variables
khst = ( (1-gst*bet*(1-del)) / (alp*gst^alp*bet) )
    ^(1/(alp-1));
xist = ( ((khst*gst)^alp - (1-gst*(1-del))*khst)/mst )
    ^(-1);
nust = psi*mst^2/( (1-alp)*(1-psi)*bet*gst^alp*khst^
    alp );
n = xist/(nust+xist);
P = xist + nust;
k = khst*n;
l = psi*mst*n/( (1-psi)*(1-n) );
c = mst/P;
d = l - mst + 1;
y = k^alp*n^(1-alp)*gst^alp;
R = mst/bet;
```

Methods of finding the steady state

```
// You can use MATLAB functions which return several
    arguments
[W, e] = my_function(l, n);
gp_obs = m/dA;
gy_obs = dA;
end;
steady;
```

- MATLAB function can be directly used in steady_state_model to obtain steady states of some particular endogenous variables

Methods of finding the steady state

```
// You can use MATLAB functions which return several
    arguments
[W, e] = my_function(l, n);
gp_obs = m/dA;
gy_obs = dA;
end;
steady;
```

- MATLAB function can be directly used in steady_state_model to obtain steady states of some particular endogenous variables

Contents

Model descrtiption

- Consider the basic Real Business Cycle (RBC) model with leisure. The representative household maximizes present as well as expected future utility.

$$\max \quad E_t \sum_{j=0}^{\infty} \beta^j U_{t+j} \tag{1}$$

- with $\beta < 1$ denonting the discount factor and $E_t$is expectation given information at time t

Model descrtiption

- define the budget constraint of the household as follow:

$$C_t + I_t = W_t L_t + R_t K_t + \Pi_t \tag{2}$$

- The law of motion for capital K_t at the end of period t is given by

$$K_t = (1 - \delta) K_{t-1} + I_t \tag{3}$$

  δ is depreciation rate

- Productivity A_t is the driving force of the economy and evolves according to

$$\log A_t = \rho_A \log A_{t-1} + \varepsilon_t^A \tag{4}$$

  where ρ_A is the persistence parameters and ε_t^A is assumed to be normally distributed with mean zero and variance σ^2.

Model descrtiption

- Real profit Π_t of the representative firm are revenues from selling output Y_t minus costs from labor $W_t L_t$ and renting capital $R_t K_{t-1}$

$$\Pi_t = Y_t - W_t L_t - R_t K_{t-1} \tag{5}$$

- The representative firm maximizes expected profits

$$\Pi_t = Y_t - W_t L_t - R_t K_{t-1} \tag{6}$$

subject to a Cobb-Douglas production function

$$f\left(K_{t-1}, L_t\right) = Y_t = A_t K_{t-1}^{\alpha} L_t^{1-\alpha} \tag{7}$$

Model descrtiption

- labor and goods market are clear in equilibrium.

$$Y_t = C_t + I_t \tag{8}$$

First-order condtions

- the first-order conditions of the representative household are given by

$$U_t^c = \beta E_t \left[U_{t+1}^c \left(1 - \delta + R_{t+1} \right) \right] \tag{9}$$

$$W_t = -\frac{U_t^L}{U_t^C} \tag{10}$$

First-order condtions

- The Lagrangian for the household problem is:

$$
\begin{aligned}
L = & \; E_t \sum_{j=0}^{\infty} \beta^j U_{t+j}\left(C_{t+j}, L_{t+j}\right) \\
& + \beta^j \lambda_{t+j}\left[W_{t+j} L_{t+j} + R_{t+j} K_{t-1+j} - C_{t+j} - I_{t+j}\right] \\
& + \beta^j \mu_{t+j}\left[(1-\delta) K_{t-1+j} + I_{t+j} - K_{t+j}\right]
\end{aligned} \tag{11}
$$

First-order condtions

- The first order condition C_t is given by

$$\frac{\partial L}{\partial C_t} = E_t\left(U_t^C - \lambda_t\right) = 0 \tag{12}$$

- The first order condition L_t is given by

$$\frac{\partial L}{\partial L_t} = E_t\left(U_t^L + \lambda_t W_t\right) = 0 \tag{13}$$

- The first order condition I_t is given by

$$\frac{\partial L}{\partial I_t} = E_t \beta^j \left(-\lambda_t + \mu_t\right) = 0 \tag{14}$$

- The first order condition K_t is given by

$$\frac{\partial L}{\partial K_t} = E_t\left(-\mu_t\right) + E_t \beta\left(\lambda_{t+1} R_{t+1} + \mu_{t+1}\left(1-\delta\right)\right) = 0 \tag{15}$$

First-order condtions

- (12) and (14) in (15) yields

$$U_t^c = \beta E_t \left[U_{t+1}^c \left(1 - \delta + R_{t+1} \right) \right] \tag{16}$$

This is the Euler equation of intertemporal optimality. It reflects the trade-off between consumption and savings.

- (12) in (13) yields

$$W_t = -\frac{U_t^L}{U_t^C} \tag{17}$$

the real wage must be equal to the marginal rate of substitution between labor and consumption.

First-order condtions

- Firm's objective is to maximize profits

$$\Pi_t = A_t K_{t-1}^{\alpha} L_t^{1-\alpha} - W_t L_t - R_t K_{t-1} \tag{18}$$

- The first-order conditions are given by:

$$\frac{\partial \Pi_t}{\partial L_t} = (1-\alpha)\frac{Y_t}{L_t} \tag{19}$$

The real wage must be equal to the marginal product of labor.

$$\frac{\partial \Pi_t}{\partial K_{t-1}} = \alpha \frac{Y_t}{K_{t-1}} \tag{20}$$

The real interest rate must be equal to the marginal product of capital.

Compute steady state

- The steady state of this model is a fixed point. there is a set of values for the endogenous variables that in equilibrium and in the absence of shocks remain constant over time.

$$\log \bar{A} = 0 \Leftrightarrow \bar{A} = 1 \tag{21}$$

- The Euler equation in steady state becomes:

$$\bar{R} = \alpha \bar{A} \bar{K}^{\alpha-1} \bar{L}^{1-\alpha} \tag{22}$$

$$\frac{\bar{K}}{\bar{L}} = \left(\frac{\alpha \bar{A}}{\bar{R}} \right)^{\frac{1}{1-\alpha}} \tag{23}$$

Compute steady state

- The firms demand for labor in steady state becomes

$$W = (1-\alpha)\,\bar{A}\bar{K}^{\alpha}\bar{L}^{1-\alpha} \tag{24}$$

- The production function in steady state becomes

$$\frac{\bar{Y}}{\bar{L}} = \bar{A}\left(\frac{\bar{K}}{\bar{L}}\right)^{\alpha} \tag{25}$$

- The clearing of the goods market in steady state implies

$$\frac{\bar{C}}{\bar{L}} = \frac{\bar{Y}}{\bar{L}} - \frac{\bar{I}}{\bar{L}} = \frac{\bar{Y}}{\bar{L}} - \delta\frac{\bar{K}}{\bar{L}} \tag{26}$$

Compute steady state

- if the utility function is given by

$$U_t = \gamma \frac{C_t^{1-\eta_c} - 1}{1-\eta_c} + \psi \frac{(1-L_t)^{1-\eta_L} - 1}{1-\eta_L} \tag{27}$$

- we can derive a closed-form expression:

$$\psi \frac{1}{1-\bar{L}} = \gamma \bar{C}^{-1} W \tag{28}$$

$$\bar{L} = \frac{\frac{\gamma}{\psi}\left(\frac{\bar{C}}{\bar{L}}\right)^{-1} W}{1 + \frac{\gamma}{\psi}\left(\frac{\bar{C}}{\bar{L}}\right)^{-1} W} \tag{29}$$

- it is straigforward to compute the remaining steady state values

$$\bar{C} = \frac{\bar{C}}{\bar{L}}\bar{L}, \bar{I} = \frac{\bar{I}}{\bar{L}}\bar{L}, \bar{K} = \frac{\bar{K}}{\bar{L}}\bar{L}, \bar{Y} = \frac{\bar{Y}}{\bar{L}}\bar{L} \tag{30}$$

Compute steady state

- if the utility function is given by

$$U_t = \gamma \log \left(C_t \right) + \psi \log \left(1 - L_t \right) \tag{31}$$

- The steady state for labor changes to

$$W \left(\frac{\bar{C}}{\bar{L}} \right)^{-\eta_c} = \frac{\psi}{\gamma} \left(1 - \bar{L} \right)^{-\eta_L} \bar{L}^{\eta_c} \tag{32}$$

- This cannot be solved for L_t. an numerical optimizer can be introduced to solved forL_t.

Compute steady state

- Introduce different ways to compute steady state.
  method 1: steady_state_model

```
var Y C K L A R W I;
varexo eps_A;
parameters alph betta delt gam pssi rhoA;
alph = 0.35; betta = 0.99; delt = 0.025; gam = 1; pssi
    = 1.6; rhoA = 0.9;

model;                          \\Log-utility
    #UC  = gam*C^(-1);
    #UCp = gam*C(+1)^(-1);
    #UL  = -pssi*(1-L)^(-1);
    UC = betta*UCp*(1-delt+R(+1));
    W = -UL/UC;
    K = (1-delt)*K(-1)+I;
    Y = I+C;
    Y = A*K(-1)^alph*L^(1-alph);
    W = (1-alph)*Y/L;
```

Compute steady state

```
R = alph*Y/K(-1);
    log(A) = rhoA*log(A(-1))+eps_A;
end;
steady_state_model;
    A = 1;
    R = 1/betta+delt-1;
    K_L = ((alph*A)/R)^(1/(1-alph));
    W = (1-alph)*A*K_L^alph;
    I_L = delt*K_L;
    Y_L = A*K_L^alph;
    C_L = Y_L-I_L;
    % closed-form expression for labor when using log
       utility
    L = gam/pssi*C_L^(-1)*W/(1+gam/pssi*C_L^(-1)*W);
    C = C_L*L;
    I = I_L*L;
    K = K_L*L;
    Y = Y_L*L;
end;
```

Compute steady state

- method 2: steady_state_model with helper function

```
var Y C K L A R W I;
varexo eps_A;
parameters alph betta delt gam pssi rhoA etaC etaL;
alph = 0.35; betta = 0.99; delt = 0.025; gam = 1; pssi
    = 1.6; rhoA = 0.9; etaC  = 2; etaL  = 1.5;
model;                          \\CES-utility
    #UC  = gam*C^(-etaC);
    #UCp = gam*C(+1)^(-etaC);
    #UL  = -pssi*(1-L)^(-etaL);
    UC = betta*UCp*(1-delt+R(+1));
    W = -UL/UC;
    K = (1-delt)*K(-1)+I;
    Y = I+C;
    Y = A*K(-1)^alph*L^(1-alph);
    W = (1-alph)*Y/L;
    R = alph*Y/K(-1);
    log(A) = rhoA*log(A(-1))+eps_A;
end;
```

Compute steady state

```
steady_state_model;
    A = 1;
    R = 1/betta+delt-1;
    K_L = ((alph*A)/R)^(1/(1-alph));
    W = (1-alph)*A*K_L^alph;
    I_L = delt*K_L;
    Y_L = A*K_L^alph;
    C_L = Y_L-I_L;
    % closed-form expression for labor is not possible
        , so we need a helper function
    L0 = 1/3;
    L = rbc_ces1_steadystate_helper(L0,pssi,etaL,etaC,
        gam,C_L,W);
    C = C_L*L;
    I = I_L*L;
    K = K_L*L;
    Y = Y_L*L;
end;
```

Compute steady state

- In the steady_state_model block,we are calling the helper function rbc_ces1_steadystate_helper.m,which is needed to be created in MATLAB.

```
function L = rbc_ces1_steadystate_helper(L0,pssi,
    etaL,etaC,gam,C_L,W)
    if etaC == 1 && etaL == 1
        L = gam/pssi*C_L^(-1)*W/(1+gam/pssi*C_L
            ^(-1)*W);
    else
        options = optimset('Display','Final','TolX
            ',1e-10,'TolFun',1e-10);
        L = fsolve(@(L) pssi*(1-L)^(-etaL)*L^etaC
            - gam*C_L^(-etaC)*W, L0,options);
    end
end
```

Contents

Deterministic simulation

- In deterministic simulation, The purpose of the simulation is to describe the reaction to the shocks, until the system returns to the old or to a new state of equilibrium.

```
Command: perfect_foresight_setup ;
```

- Computes the perfect foresight (or deterministic) simulation of the model.

```
perfect_foresight_solver;
```

Note that perfect_foresight_setup must be called before this command, in order to setup the environment for the simulation.

Stochastic simulation

- In a stochastic context, Dynare computes one or several simulations corresponding to a random draw of the shocks.
- Computing the stochastic solution

```
Command: stoch_simul [VARIABLE_NAME...];
```

- stoch_simul computes a Taylor approximation of the model around the deterministic steady state and solves of the the decision and transition functions for the approximated model.

Contents

Summary

- Dynare is a quite useful toolbox for beginners in the study of Dynamics.
- Knowing functions of different blocks is the most important thing for learners.
- It is necessary to be familiar with different ways of computing steady states, which is the most challenging part of using dynare to solve dynamic models.
- Note that when confronted with quiet complicated dynamic models or some specific problems, Dynare may not be useful.