Dynamic Macroeconomic Models

Lecture 3

Alessandro Di Nola

University of Konstanz

Outline

- How to install and run Dynare
- Introduction to Dynare
- Incorporating Dynare in other Matlab programs
- Peculiarities of Dynare and advanced tricks
- Impulse Response Functions
- Stochastic simulations

References

- Of course, Dynare's reference manual:
  - Adjemian, S. et al., 2011. "Dynare: Reference Manual Version 4," Dynare Working Papers 1, CEPREMAP, revised Jul 2018.
- A good textbook treatment is provided in: Miao, J., 2014 "Economic Dynamics in Discrete Time," MIT Press. Especially chapters 2.5 and 15.
- Lots of material on the web about Dynare
  - `https://forum.dynare.org/`

How to install and run Dynare

To install it:

- Go to the Dynare website: `https://www.dynare.org/`.
- In the main menu, go to "Download".
- Download current stable release
  - Available for Windows, Mac, Linux.
- Run the executable. It will install Dynare in your C drive.
- Start Matlab, click in the file menu on "set path".
- Click on the button "Add folder". Warning: DO NOT select "Add with Subfolders".
- Select the matlab subdirectory of your Dynare installation. For ex., select:

  ```
  C:\dynare\4.5.7\matlab
  ```

- Apply the setting by clicking "Save" button.

How to install and run Dynare

How to run it:

- In the Matlab main window, change the directory to the one where you have stored your Dynare model file.
- To run the program `myfile.mod`, type the command:

  `dynare myfile.mod`
- You can type the above command either in the Matlab command window or in a Matlab script.
- Warning: Dynare clears all variables out of memory. To overrule this, add option "noclearall":

  `dynare myfile.mod noclearall`

Stochastic growth model, with Dynare

- Same stochastic growth model as in Lecture 2, planner maximizes

$$\mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\sigma}}{1-\sigma}$$

subject to resource constraints

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

given productivity

$$\log z_{t+1} = \phi \log z_t + \varepsilon_{t+1}, \quad \varepsilon_{t+1} \sim \text{IID } N\left(0, \sigma_{\varepsilon}^2\right)$$

for some given parameters $\alpha, \beta, \delta, \varphi, \sigma, \sigma_{\varepsilon}$ (here $\bar{z} = 1$).

Dynare model file

The structure of a Dynare model file (`.mod`)

1. Declare endogenous variables
2. Declare exogenous variables
3. Declare parameters
4. Declare the model equations
5. Ask Dynare to solve for the steady state
6. Ask Dynare to solve for the the dynamics

(1) Declare endogenous variables

- From `stochastic_growth_dynare.mod` in ILIAS

```
// (1) declare endogenous variables
var c, k, z;
```

- Note productivity z_t is treated as endogenous.

(2) Declare exogenous variables

- It is the innovations ε that are fundamentally exogenous

```
// (2) declare exogenous variables (shocks)
varexo e;
```

(3) Declare parameters

- List of parameter names

```
// (3) declare parameters
parameters alpha, beta, delta, sigma, rho, sigmaeps;
```

- Set parameter values

```
alpha    = 0.35;
beta     = 0.95;
delta    = 0.1;
rho      = 0.95;
sigma    = 3.00;
sigmaeps = 0.01;
```

Dynare notation for model equations

Resource constraint

- In usual notation

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

- In Dynare notation, *supposing we want an approximation in logs*

```
exp(c) + exp(k)
= exp(z)*(exp(k(-1))^alpha)+(1-delta)*exp(k(-1));
```

  (otherwise approximation will be in levels)
- Dynare timing convention:
  - Variables chosen at t have no time argument
  - Variables chosen at $t-1$ have -1 argument
  - Variables chosen at $t+1$ have $+1$ argument

Dynare notation for model equations

Consumption Euler equation

- In usual notation:

$$c_t^{-\sigma} = \beta \mathbb{E}_t \left\{ c_{t+1}^{-\sigma} \left[z_{t+1} \alpha k_{t+1}^{\alpha-1} + 1 - \delta \right] \right\}$$

- In Dynare notation, supposing we want an approximation in logs

```
exp(c)^(-sigma)
= beta*(exp(c(+1))^(-sigma))
*(alpha*exp(z(+1))*(exp(k)^(alpha-1))+1-delta);
```

Dynare notation for model equations

AR(1) for productivity shock

- In usual notation

$$\log z_{t+1} = \rho \log z_t + \varepsilon_{t+1}$$

- In Dynare notation, supposing we want an approximation in logs

```
z = rho*z(-1) + e;
```

(4) Declare the model equations

- Start block with **model**, then list equations, then **end**

```
model;
// resource constraint
exp(c) + exp(k)
= exp(z)*(exp(k(-1))^alpha)+(1-delta)*exp(k(-1));
// consumption Euler equation
exp(c)^(-sigma)
= beta*(exp(c(+1))^(-sigma))
*(alpha*exp(z(+1))*(exp(k)^(alpha-1))+1-delta);
// law of motion productivity
z = rho*z(-1) + e;
end;
```

(5) Solve for the steady state

- Solve for steady state numerically (system of nonlinear equations)
- Start block with **initval**, then list guess, then **end**

```
// (5) solve the steady state
initval;
c = 0.75;
k = 3;
z = 1;
e = 0;
end;
steady;
```

- Can enter exact solution here if you have one
- Given model equations that we declared, steady-state is in *logs*.

Set the shocks

- Set the variance/covariance structure of shocks

```
// specify variance of shocks
shocks;
var e = 100*sigmaeps^2;
end;
```

- Only one shock here so this is simple.
- Later: example with two correlated shocks.

(6) Solve for the dynamics

- Solve for coefficients, obtain moments, plot impulse responses, etc.

  ```
  // (6) solve the dynamics
  stoch_simul(order=1,irf=100);
  ```

- Lots of options available (see user guide for details).
- Order of the Taylor approximation: `order = 1,2,3`.
- Note: the default is `order = 2`.

(6) Solve for the dynamics

Impulse responses and simulations

- Setting `irf = 0` suppresses the plotting of IRFs (useful if you nest Dynare into another Matlab program):

  ```
  stoch_simul(order=1,IRF=0)
  ```

- Don't plot IRFs, don't print the correlation matrix, don't print moments of the endo variables:

  ```
  stoch_simul(order=1,nocorr,nomoments,irf=0);
  ```

- Don't print the graphs. Useful for loops.

  ```
  stoch_simul(order=1,nograph,irf=100);
  ```

- Don't print anything. Useful for loops.

  ```
  stoch_simul(order=1,noprint,irf=100);
  ```

Dynare output

Steady-state results

```
STEADY-STATE RESULTS:

c    0.186403
k    1.27678
z    0
```

- Note: Given model equations we have declared, steady state is in logs.

Dynare output

Policy and transition functions

POLICY AND TRANSITION FUNCTIONS

	c	k	z
Constant	0.186403	1.276779	0
k(-1)	0.382458	0.924091	0
z(-1)	0.676055	0.187070	0.950000
e	0.711637	0.196916	1.000000

- Policy functions produced by Dynare are slightly different from what we are used to.
- Dynare artificially splits z_t into predetermined component and an exogenous component.

Dynare output

Policy and transition functions

```
POLICY AND TRANSITION FUNCTIONS
              c             k               z
Constant      0.186403      1.276779               0
k(-1)         0.382458      0.924091               0
z(-1)         0.676055      0.187070        0.950000
e             0.711637      0.196916        1.000000
```

- What we are used to see:

$$\tilde{c}_t = \psi_{ck} \tilde{k}_{t-1} + \psi_{cz} \tilde{z}_t$$

- What Dynare spits out:

$$\log c_t = \log \overline{c} + \psi_{ck} (\log k_{t-1} - \log \overline{k}) + (\psi_{cz} \rho) \log z_{t-1} + \psi_{cz} \varepsilon_t$$

Dynare output

Theoretical moments

THEORETICAL MOMENTS

VARIABLE	MEAN	STD. DEV.	VARIANCE
c	0.1864	0.4458	0.1987
k	1.2768	0.6471	0.4187
z	0.0000	0.3203	0.1026

Dynare output

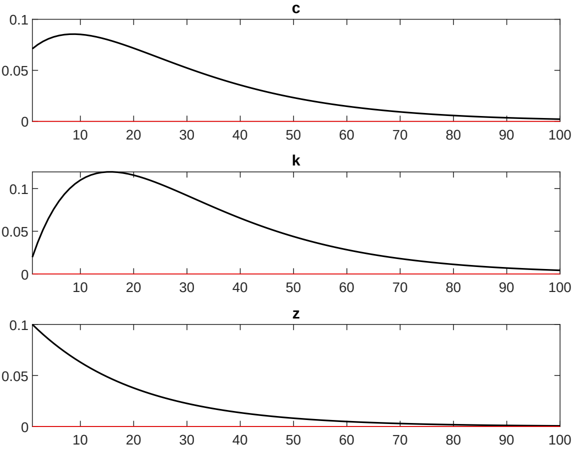
Correlations

MATRIX OF CORRELATIONS

Variables	c	k	z
c	1.0000	0.9621	0.9322
k	0.9621	1.0000	0.7981
z	0.9322	0.7981	1.0000

Dynare output

Impulse responses



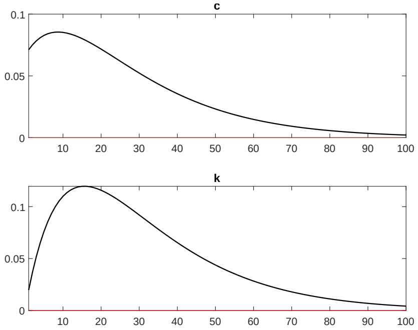
- IRF of c, k and z to a 1 st.dev. shock to ε.

Dynare output

Impulse responses, cont'd

- By default, Dynare plots IRFs of all endogenous variables.
- What if you are interested only in a subset of them?
    - E.g. only consumption and capital, not technology
- Change `stoch_simul` command as follows:

```
stoch_simul(order=1,irf=100) c k;
```



Dynare output

Matlab workspace

- Where does Dynare store results?

Name	Size	Bytes	Class	Attributes
M_	1x1	8280	struct	global
alpha	1x1	8	double	
beta	1x1	8	double	
c_e	150x1	1200	double	
delta	1x1	8	double	
estimation_info	1x1	33616	struct	global
ex0_	0x0	0	double	global
info	1x1	8	double	
k_e	150x1	1200	double	
oo_	1x1	10768	struct	global
options_	1x1	74358	struct	global
phi	1x1	8	double	
sigma	1x1	8	double	
sigmaeps	1x1	8	double	
var_list_	0x0	0	double	
ys0_	0x0	0	double	global
z_e	150x1	1200	double	

Dynare output

Matlab workspace

- Parameters that we supplied:

  `alpha, beta, delta, rho, sigma, sigmaeps`

- Impulse responses of each endogenous variable to each shock:

  `c_e, k_e, z_e`

- Specialized Matlab structures that store key results:

  `M_, options_, oo_, ...`

- These structures are also saved in `stochastic_growth_dynare_results.mat` created by running Dynare on `stochastic_growth_dynare.mod`.

Dynare output

Structures

- Structures created by Dynare
  - `M_`, information about the model
  - `options_`, options used during computation
  - `oo_`, results of the computation
- Get structure entries by "`structure_.entry`", for example

  `oo_.steady_state`

  recovers the steady state values given on a previous slide.
- Warning: Structure entries can be relatively complicated objects!!

Dynare output

Structures

- For example, the entries of `oo_` are

```
         exo_simul: [3x1 double]
        endo_simul: []
                dr: [1x1 struct]
   exo_steady_state: 0
exo_det_steady_state: []
      exo_det_simul: []
       steady_state: [3x1 double]
            gamma_y: {6x1 cell}
               mean: [3x1 double]
                var: [3x3 double]
           autocorr: {1x5 cell}
               irfs: [1x1 struct]
```

Dynare output

Structures

- Structures can be nested into each other.
- Here, `oo_.irfs` is another structure!
- If you type `oo_.irfs` in the command window...

```
 struct with fields:
c_e: [1*100 double]
k_e: [1*100 double]
z_e: [1*100 double]
```

Dynare output

Decision rules structures

- Where does Dynare store the matrices of the state-space form solution?

```
POLICY AND TRANSITION FUNCTIONS
              c                 k                 z
Constant      0.186403          1.276779                 0
k(-1)         0.382458          0.924091                 0
z(-1)         0.676055          0.187070          0.950000
e             0.711637          0.196916          1.000000
```

Dynare output

Decision rules structures

- Where does Dynare store the matrices of the state-space form solution?
- `oo_.dr.ghx` is the 3×2 matrix:

0.9241	0.1871
0	0.9500
0.3825	0.6761

- while `oo_.dr.ghu` is the 3×1 matrix

```
0.1969
1.0000
0.7116
```

Dynare output

Decision rules structures

- To simplify, let `oo_.dr.ghx` be g_x and `oo_.dr.ghu` be g_u.
- From the above, can see that Dynare gives the state-space solution in the following format:

$$\begin{bmatrix} k_t \\ z_t \\ c_t \end{bmatrix} = \underbrace{\begin{bmatrix} 0.9241 & 0.1871 \\ 0 & 0.95 \\ 0.3825 & 0.6761 \end{bmatrix}}_{g_x} \begin{bmatrix} k_{t-1} \\ z_{t-1} \end{bmatrix} + \underbrace{\begin{bmatrix} 0.1969 \\ 1 \\ 0.7116 \end{bmatrix}}_{g_u} \varepsilon_t$$

- Even if in the `*.mod` file we ordered the variables as "c, k, z", Dynare orders them as "k, z, c".
- For policy functions matrices g_x and g_u unfortunately Dynare does NOT follow the order of declaration chosen by the user, but the so-called DR-order.

Dynare output

Decision rules structures, cont'd

- Declaration order chosen by the user:
  - In our example, the declaration order is "c, k, z".
- DR-order (DR stands for decision rules):
  - Static variables
  - State/backward looking variables
  - Control/forward looking variables
- Within each category, variables are arranged according to the declaration order.
  - In our example, the DR-order is "k, z, c". (There are no static variables).
- More on this and other advanced Dynare topics in the next class.

Dynare: Advanced features

Peculiarities of Dynare: timing convention

- All variables known at time t must be dated $t-1$ (i.e. k_t is a choice variable, k_{t-1} is a state or predetermined variable).
- This is different from convention used in most of the macro literature.
- Can change this timing convention using the `predetermined_variables` command.
- The following two examples are exactly equivalent.

Peculiarities of Dynare: timing convention, cont'd

Using default Dynare timing convention:

```
var y, k, i;
...
model;
y = k(-1)^alpha;
k = i + (1-delta)*k(-1);
...
end;
```

Using the alternative timing convention:

```
var y, k, i;
predetermined_variables k;
...
model;
y = k^alpha;
k(+1) = i + (1-delta)*k;
...
end;
```

Adding extra variables

- Suppose you would like to include also output and interest rate in your analysis.
- Modify the `*.mod` file as follows.
- Change (1) by adding the extra variables

```
// (1) declare endogenous variables
    var  c, k, y, r, z;
```

- Change model equations in (4) by adding

```
// Output
y = exp(z)*(exp(k(-1))^alpha);
// Interest rate
r = alpha*exp(z)*(exp(k(-1))^(alpha-1));
```

Adding extra variables, cont'd

- Finally, you need to provide initial conditions in the steady-state block (5) also for the new variables:

```
initval;
c = 0.75;
k = 3;
z = 1;
e = 0;
y = exp(3)^alpha;
r = alpha*(exp(3)^(alpha-1));
end;
```

Linear model

- Suppose you have already log-linearized your model equations.
- Or you prefer to work with linearized solution for any reason
  - Common for the three-equations New-keynesian model.
- Then you should add option "linear" to your model block declaration:

```
model(linear);


end;
```

- The option "linear" tells Dynare your model is already linear and thus speeds up some computations.
- However, when your model is already linear, not putting "linear" does of course not influence the results.
  - A linear approximation of a linear equation is simply the equation itself!

Loops over parameters

- This trick allows you to run the same Dynare program for different parameter values.
- Suppose your Dynare program has the command

  `rho=0.8;`
- You would like to run the program twice; once for $\rho = 0.8$, and once for $\rho = 0.9$.
- Of course you could write two separate `*.mod` files, one for each parameter value.
- But there is a better solution...

Loops over parameters, cont'd

1. In your Matlab program, loop over the different values of rho. Save the value of rho and the associated name to the file "parameterfile":

   ```
   save parameterfile rho
   ```

   and *then* run Dynare.

2. In your Dynare program file, replace the command `rho=0.8` with:

   ```
   load parameterfile
   set_param_value('rho',rho);
   ```

- Remarks:
  - The name of the file is arbitrary.
  - In `set_param_value('.',.)`, the first argument is the name in your *.mod file and the second is the numerical value.