Solving and Simulating DSGE Models with Dynare

Based on Notes by Eric Sims, University of Notre Dame (Spring 2024)

Macroeconomics II - Universidad Alberto Hurtado

May 12, 2025

Introduction to Dynare

- What is Dynare?
  - A tool for solving, simulating, and estimating DSGE models.
  - A collection of MATLAB codes, not a standalone program.
- Installation:
  - Download from *`http://www.dynare.org/download`*.
  - Requires MATLAB.
  - Example path: *`C:\dynare\4.4.3\matlab`*.
- Running Dynare:
  - Create *`.mod`* files.
  - In MATLAB: *`dynare filename`*.
  - Or, set MATLAB path.

State-Space Representation (1st Order)

Dynare solves the model into a state-space form:

- State transition: $s_t = As_{t-1} + B\epsilon_t$
- Observation/Policy function: $x_t = \Phi s_t$

Dynare often presents it by substituting states into the policy function:

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

Or more compactly:

$$s_t = As_{t-1} + B\epsilon_t$$

$$x_t = Cs_{t-1} + D\epsilon_t$$

where $C = \Phi A$ and $D = \Phi B$. Combined: $Y_t = \Psi s_{t-1} + \Omega \epsilon_t$ where

$Y_t = [s_t' \; x_t']'$, $\Psi = [A' \; C']'$, $\Omega = [B' \; D']'$.

Example: Neoclassical Model with Fixed Labor

- Planner's Problem:

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

  Subject to:

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

- TFP Process (AR(1) in logs):

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

- Parameter Calibration Example:
    - $\sigma = 1$, $\alpha = 1/3$, $\delta = 0.025$
    - $\beta = 0.99$, $\rho = 0.95$
    - std(productivity shock) = 0.01.

Model Equations: First Order Conditions (FOCs)

- Euler Equation:

$$c_t^{-\sigma} = \beta E_t c_{t+1}^{-\sigma}(\alpha a_{t+1} k_{t+1}^{\alpha-1} + (1-\delta))$$

- Resource Constraint:

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

- TFP Process:

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

Model Equations: Auxiliary Variables

- Auxiliary Variables (Output, Investment):

$$y_t = a_t k_t^{\alpha}$$

$$i_t = y_t - c_t$$

- Prices (Real Interest Rate r_t, Rental Rate R_t, Wage w_t):

$$c_t^{-\sigma} = \beta E_t c_{t+1}^{-\sigma}(1 + r_t)$$

$$R_t = \alpha a_t k_t^{\alpha-1}$$

$$w_t = (1 - \alpha) a_t k_t^{\alpha}$$

Dynare Timing Convention

- **Rule:** Predetermined variables (e.g., capital k) appear as $t-1$ in time t equations, and t in $t+1$ equations. This means k_t in the model becomes k_{t-1} in Dynare if it's predetermined from last period, and k_{t+1} in the model becomes k_t in Dynare.
- **Rewritten FOCs (Examples):**
    - Euler: $c_t^{-\sigma} = \beta E_t c_{t+1}^{-\sigma}(\alpha a_{t+1} k_t^{\alpha-1} + (1-\delta))$
    - Constraint: $k_t = a_t k_{t-1}^{\alpha} - c_t + (1-\delta)k_{t-1}$
    - Output: $y_t = a_t k_{t-1}^{\alpha}$

  (Essentially, lag capital one period in relevant equations compared to the model's original timing.)

Writing Dynare Code: Declarations

1. Endogenous Variables (‘var‘):

```
var y I k a c w R r;
```

2. Exogenous Variables/Shocks (‘varexo‘):

```
varexo e;
```

Note: Each entry must end with a semicolon.

3. Parameters (‘parameters‘):

```
parameters alpha beta delta rho sigma sigmae;
```

4. Assign Parameter Values:

```
    alpha = 1/3;
    beta = 0.99;
    delta = 0.025;
    rho = 0.95;
    sigma = 1;
    sigmae = 0.01;
```

Note: Each entry must end with a semicolon.

Writing Dynare Code: Model Block

- Structure: `model;` … equations … `end;`.
- **Log-Linearization:** For approximation of logs, write variables as `exp(x)`. `x` is then $\ln(X)$.
  - We want to generate impulse response functions that we can interpret as percentage deviations.
- **Timing:** $x_t \rightarrow x$, $x_{t+1} \rightarrow x(+1)$, $x_{t-1} \rightarrow x(-1)$.

Model Block:

```
model;
exp(c)^(-sigma) = beta*exp(c(+1))^(-sigma)*(alpha*exp(a
    (+1))*exp(k)^(alpha-1)+(1-delta));
exp(y) = exp(a)*exp(k(-1))^(alpha);
exp(k) = exp(a)*exp(k(-1))^(alpha) - exp(c) + (1-delta)*
    exp(k(-1));
a = rho*a(-1) + e;
exp(y) = exp(c) + exp(I);
exp(c)^(-sigma) = beta*exp(c(+1))^(-sigma)*(1+r);
exp(R) = alpha*exp(a)*exp(k(-1))^(alpha-1);
exp(w) = (1-alpha)*exp(a)*exp(k(-1))^(alpha);
end;
```

Writing Dynare Code: Initial Values & Shocks

Initial Values for Steady State (‘initval‘):

- Dynare numerically solves for steady state, needs guesses.
- If using *exp(x)*, initial values for x (i.e., $\ln(X_{ss})$).

```
initval;
k = log(30);
y = log(3);
c = log(2.5);
I = log(0.5);
a = 0;
r = (1/beta)-1;
R = log((1/beta)-(1-delta));
w = log(1);
end;
```

Writing Dynare Code: Initial Values & Shocks

Shock Variance (‘shocks‘): Specify variance (not std. dev.).

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

Calculate Steady State:

```
steady;
```

Solve and Simulate: Solves, Produces Policy Functions, IRFs, Moments

```
stoch_simul(options);
```

Solving and Simulating

Common Options:

- *order=1*: First-order (linear) approximation (default is 2nd).
- *irf=integer*: Number of periods for IRFs (default 40). *irf=0* suppresses.
- *hp_filter=integer*: Produces HP-filtered moments (e.g., 1600).
- *periods=integer*: Simulate data and take moments from simulation.
- *nofunctions*, *nomoments*, *noprint*.

Example:

```
stoch_simul(order=1, irf=20);
```

Running Dynare & Helper .m Files

Execution Command: *`dynare filename`* (without .mod).

- *`dynare filename noclearall`*: Prevents clearing workspace.
- *`dynare filename nolog`*: Prevents log file creation.

Running Dynare & Helper .m Files

Accompanying .m File for Automation:

```
clear all; close all;
% specify parameters
beta = 0.99; alpha = 1/3; sigma = 1;
sigmae = 0.01; rho = 0.9; delta = 0.025;
save param_nc alpha beta delta rho sigma sigmae;
dynare basic_nc_dynare_alt noclearall nolog;
```

Running Dynare and Helper .m Files

Modifying .mod to Load Parameters: Use 'load' and '`set_param_value`'.

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

Custom Steady-State File (_steadystate.m)

- Purpose: Analytically solve for steady state.
- Naming: `your_mod_filename_steadystate.m`.
- Function def: `function [ys,check] = ..._steadystate(ys_init,M_)`
- Access parameters: `M_.params`. Store results in `ys`. `check=0`.

Custom Steady-State File (_steadystate.m)

Example Code Snippet (`basic_nc_dynare_ss_steadystate.m`):

```
function [ys,check] = basic_nc_dynare_ss_steadystate(ys_init,M_)
global M_; // Not strictly needed
alpha = M_.params(1); beta = M_.params(2); // ... and so on
k_ss = (alpha / ( (1/beta) - (1-delta_val) ) )^(1/(1-alpha));
y_ss = k_ss^(alpha);
I_ss = delta*k_ss;
a_ss = 1; // Or 0 if log(a)
// ... other steady state values
// Assign to ys (in logs if needed, in order of var declaration)
ys = [log(y_ss); log(I_ss); ... ]; // Complete for all vars, remember to use a_ss if using log(a)
check = 0;
```

Custom Steady State

```
function [ys,check] = basic_nc_dynare_ss_steadystate(ys,exe);

global M_

alpha = M_.params(1);
beta = M_.params(2);
Δ = M_.params(3);
rho = M_.params(4);
sigma = M_.params(5);
sigmae = M_.params(6);

k = (alpha/(1/beta - (1-Δ)))^(1/(1-alpha));
y = k^(alpha);
I = Δ*k;
c = y - I;
a = 1;
w = (1-alpha)*k^(alpha);
R = alpha*k^(alpha-1);
r = (1/beta) - 1;

check = 0;

ys = [log(y);
    log(I);
    log(k);
    log(a);
    log(c);
    log(w);
    log(R);
    r];
```

Dynare Output: Steady State

- STEADY STATE
- MODEL SUMMARY
- VARIANCE-COVARIANCE MATRIX
- POLICY AND TRANSITION FUNCTIONS:
    - Coefficients of (log-)linearized solution.
    - Constant term = steady state.
    - Coefficients on $k(-1), a(-1)$ and e.
- THEORETICAL MOMENTS: Mean, Std. Dev., Variance.
- MATRIX OF CORRELATIONS
- COEFFICIENTS OF AUTOCORRELATION

Dynare Output: Impulse Response Functions (IRFs)

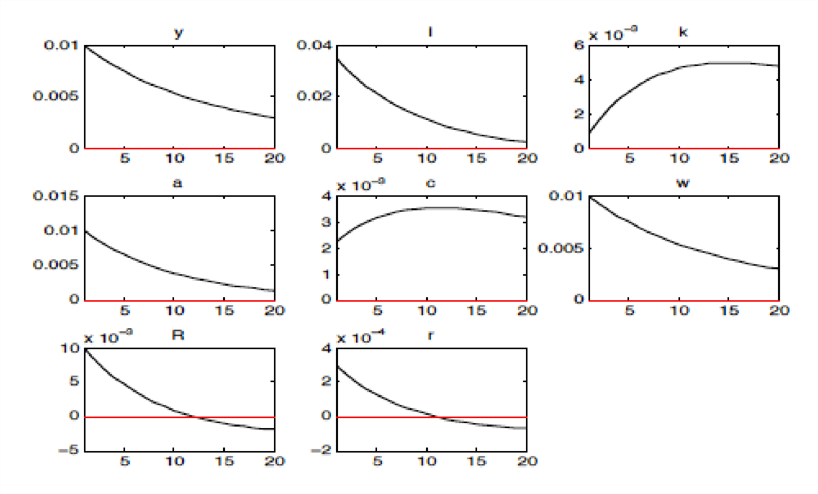

Figure 1: IRFs to a Productivity Shock

Dynare Output: Impulse Response Functions (IRFs)

Interpretation Note for Predetermined Variables (e.g., k):

- Dynare's IRF for k is for k_{t+1} (model timing).
- To get IRF of k_t (model timing):
    1. Impact response (period 0) is 0.
    2. Response of k_t at horizon h is Dynare's plotted response for k at period $h-1$ (for $h \geq 1$).

Accessing Stored Output

Results stored in structure ‘oo_‘.

- **IRFs:** ‘variablename_shockname‘ (e.g., ‘y_e‘).
- **Steady State:** ‘oo_.dr.ys‘ (order of declaration).
- **Policy/Transition Coefficients (State-Space $x_t = Cs_{t-1} + D\epsilon_t$):**
  - Matrix C (coeffs on s_{t-1}): ‘oo_.dr.ghx‘.
  - Matrix D (coeffs on ϵ_t): ‘oo_.dr.ghu‘.
- **Variable Ordering:** Stored coeffs use ”DR order”, not declaration order.
  - Mapping: ‘oo_.dr.inv_order_var‘. If var declared i-th, ‘oo_.dr.inv_order_var(i)‘ gives DR index.

Code - Defining Variable Order:

```
p_y = 1; p_I = 2; p_k = 3; p_a = 4; % ... and so on
```

References

Sims, E. (2024). Graduate Macro Theory II: Notes on Using Dynare. University of Notre Dame.

All page and snippet citations refer to this document.