Dynamic Macroeconomics using Matlab Seminar 4

University of Callao , Peru
Edinson Tolentino
email: et396@kent.ac.uk
Twitter: @edutoleraymondi

Content

Dynare model file

- The structure of a Dynare model file (`.mod`)
  - Declare endogenous variables
  - Declare exogenous variables
  - Declare parameters
  - Declare the model equations
  - Ask Dynare to solve for the steady state

Declare endogenous variables

- Note productivity z_t is treated as endogenous

Listing 1: Script

```

// (1) declare endogenous variables

var          c, k, l, z;
```

Declare exogenous variables

- It is the innovations ε_t that are fundamentally exogenous, given technology shock

Listing 2: `Script`

```

// (2) declare exogenous variables (shocks)

varexo      e;
```

Declare parameters

- List of parameter names

Listing 3: Script

```

// (3) declare parameters

parameters alpha, beta, delta, sigma, phi, sigmaeps, varphi;
```

- Set parameter values

Listing 4: Script

```

alpha    = 0.485;
beta     = 0.925;
delta    = 0.078;
phi      = 0.95;
sigma    = 1;
sigmaeps = 0.01;
varphi   = 0.397;
```

Declare parameters

- Set parameter values

Listing 5: `Script`

```

alpha    = 0.485;
beta     = 0.925;
delta    = 0.078;
phi      = 0.95;
sigma    = 1;
sigmaeps = 0.01;
varphi   = 0.397;
```

- Calibration table 1

Parameter	Meaning of parameter	Calibrate value
α	Output elasticity of capital	0.485
β	Discount factor	0.925
δ	Depreciation rate	0.078
σ	Consumption elasticity	1
z_{ss}	Technology shock	1
ρ	Autoregressive coefficient for productivity	0.95
σ_ε	Shock error	0.01
φ	Labor supply elasticity	0.397

Table: Parameter values of the structural model.

Declare the model equations

Labor elasticity

- In usual notation

$$l_t^{\varphi} c_t^{\sigma} = (1-\alpha) z_t \left(\frac{k_t}{l_t}\right)^{\alpha}$$

Listing 6: `Script`

```
// labor supply
exp(l)^(varphi)*exp(c)^(sigma) =
(1-alpha)*exp(z)*(exp(k)^(alpha))*(exp(l)^(-alpha));
```

- Variables chosen at t have no time argument
- Variables chosen at $t-1$ have -1 argument
- Variables chosen at $t+1$ have $+1$ argument

Declare the model equations

Consumption Euler equation

- In usual notation

$$c_t^{-\sigma} = \beta c_{t+1}^{-\sigma}\left[\alpha z_{t+1}\left(\frac{k_{t+1}}{l_{t+1}}\right)^{\alpha-1} + (1-\delta)\right]$$

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

Listing 7: `Script`

```
// consumption Euler equation
exp(c)^(-sigma) = beta*(exp(c(+1))^(-sigma))*
(alpha*exp(z(+1))*(exp(k(+1))^(alpha-1))*
(exp(l(+1))^(1-alpha))+1-delta);
```

- Variables chosen at t have no time argument
- Variables chosen at $t-1$ have -1 argument
- Variables chosen at $t+1$ have $+1$ argument

Declare the model equations

Resource constraint

- In usual notation

$$c_t + k_{t+1} = w_t l_t + (r_t + 1 - \delta) k_t$$

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

Listing 8: `Script`

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

- Variables chosen at t have no time argument
- Variables chosen at $t-1$ have -1 argument
- Variables chosen at $t+1$ have $+1$ argument

Declare the model equations

Law of motion for productivity

- In usual notation

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

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

Listing 9: `Script`

```
// law of motion productivity
z = phi*z(-1) + e;
```

- Variables chosen at t have no time argument
- Variables chosen at $t-1$ have -1 argument
- Variables chosen at $t+1$ have $+1$ argument

Declare the model equations

- Start block with `model` that list equations, then `end`

Listing 10: `Script`

```

// (4) declare the model equations

model;
// labor supply
exp(l)^(varphi)*exp(c)^(sigma) =
(1-alpha)*exp(z)*(exp(k)^(alpha))*(exp(l)^(-alpha));

// consumption Euler equation
exp(c)^(-sigma) = beta*(exp(c(+1))^(-sigma))*
(alpha*exp(z(+1))*(exp(k(+1))^(alpha-1))*
(exp(l(+1))^(1-alpha))+1-delta);

// resource constraint
exp(c) + exp(k) = exp(z)*(exp(k(-1))^alpha)*
(exp(l)^(1-alpha))+(1-delta)*exp(k(-1));


// law of motion productivity
z = phi*z(-1) + e;

end;
```

Solve for the steady state

- Solve for steady state numerically (system of nonlinear equations)

Listing 11: `Script`

```


// (5) solve the steady state
initval;
c = 0.75;
k = 3.5;
l = 0.3;
z = 1;
e = 0;
end;
```

Set the shocks

- Set the variance/covariance structure of shocks

Listing 12: Script

```
// specify variance of shocks

shocks;
var e = 100*sigmaeps^2;
end;
```

Solve for the dynamics

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

Listing 13: `Script`

```
// (6) solve the dynamics
stoch_simul(order=2,irf=60);
```