Dynamic Macroeconomic Models

Lecture 4

Alessandro Di Nola

University of Konstanz

Outline

- Steady-state computation
- Incorporating Dynare in matlab programs
- Blanchard-Kahn conditions and indeterminacy
- Computation of moments
    - Theoretical moments
    - Moments based on stochastic simulations

Steady-state computation

Basic

- Hardest part of Dynare is to solve for steady state.
- Solving for the steady state $\equiv$ solving a system of nonlinear equations.

Steady-state computation

Basic

- Hardest part of Dynare is to solve for steady state.
- Solving for the steady state $\equiv$ solving a system of nonlinear equations.
- Simplest way: give initial guesses using “initval” command.
- Then Dynare starts a numerical solver to find the steady-state.

```
initval;
c = 0.75;
k = 3;
z = 1;
e = 0;
end;
steady;
```

Steady-state computation

Basic

- Use economic intuition to come up with good initial guesses!
  - E.g. in the RBC model, $I < C < Y < K$.

Steady-state computation

Basic

- Use economic intuition to come up with good initial guesses!
  - E.g. in the RBC model, $I < C < Y < K$.
- Unfortunately, with more complicated models not so easy to find good guesses.
- $\rightarrow$ Basic method does not perform very well in practice.

Steady-state computation

Advanced

- Three (advanced) methods:
  1. Compute the steady state analytically and provide it to Dynare using the `steady_state_model` command.

Steady-state computation

Advanced

- Three (advanced) methods:
  1. Compute the steady state analytically and provide it to Dynare using the `steady_state_model` command.
  2. Compute the steady state numerically in Matlab and pass your numerical solution as an educated guess to Dynare.

Steady-state computation

Advanced

- Three (advanced) methods:
  1. Compute the steady state analytically and provide it to Dynare using the `steady_state_model` command.
  2. Compute the steady state numerically in Matlab and pass your numerical solution as an educated guess to Dynare.
  3. Use an explicit steady state file, which requires that you follow strict programming rules.

Steady-state computation

Advanced

- Three (advanced) methods:
  1. Compute the steady state analytically and provide it to Dynare using the `steady_state_model` command.
  2. Compute the steady state numerically in Matlab and pass your numerical solution as an educated guess to Dynare.
  3. Use an explicit steady state file, which requires that you follow strict programming rules.
- TIP: method 2 works well in practice.

Steady-state computation: Method 1

Advanced

- Stochastic growth model seen in Lectures 1/2/3.
- Deterministic steady-state:

$$\bar{k} = \left(\frac{\alpha \beta \bar{z}}{1 - \beta (1 - \delta)} \right)^{\frac{1}{1-\alpha}}, \ \bar{c} = \bar{z} \bar{k}^{\alpha} - \delta \bar{k}, \ \bar{z} = 1$$

- Steady-state block in Dynare:

```
steady_state_model;
z_ss = 1;
k_ss = (alpha*z_ss/(1/beta-1+delta))^(1/(1-alpha));
c_ss = z_ss*k_ss^alpha - delta*k_ss;
y_ss = z_ss*k_ss^alpha;
c = log(c_ss); k = log(k_ss);
y = log(y_ss); z = log(z_ss);
end;
steady;
```

Steady-state computation: Method 1

Advanced

- You can find the necessary files to implement this method on ILIAS. Go to `Codes`, `Dynare:steady state`.
  - Relevant files are `main1.m` and `growth1.mod`.

Steady-state computation: Method 2

Advanced

- For some models, not easy to find steady state analytically.
- Compute numerically s.s. in Matlab.
  - Use all the flexibility of Matlab programming environment.
- Save your numerical results in a *.mat file:

```
save paramfile z_ss k_ss c_ss y_ss
```

- Load *.mat file in Dynare. **Declare ss values as extra parameters**.

```
parameters alpha, beta, delta, sigma, rho, sigmaeps,
z_ss, k_ss, c_ss, y_ss;
load paramfile
set_param_value('z_ss',z_ss)
set_param_value('k_ss',k_ss)
....
```

Steady-state computation: Method 2

Advanced

- Use imported values z_ss, k_ss, c_ss, y_ss as "educated" initial conditions in the initval block.

```
initval;
c = log(c_ss);
k = log(k_ss);
y = log(y_ss);
z = log(z_ss);
end;
steady;
```

Steady-state computation: Method 2

Advanced

- How to compute numerically steady state in Matlab?
- Matlab has several nonlinear solvers.
- In practice, `fsolve` performs well.

Steady-state computation: Method 2

Advanced

- Write function with nonlinear equations that define the steady state.

  ```
  function F = ss_eq(x,params)
  ```

- Inputs: $n \times 1$ vector `x`, with n equal to # endo variables.
- Anything else in `params`.
- Ouput: vector-valued function $F : R^n \to R^n$.
- Each element of vector F is a steady state equation.
- Note: you don't have to follow *exactly* this structure.
  - E.g., can add other variables as inputs or outputs to the function.

Steady-state computation: Method 2

Advanced

```
function F = ss_eq(x,beta,alpha,delta)
c = x(1);
k = x(2);
y = x(3);
z = x(4);
% Pre-allocate vector F(x)
F = zeros(4,1);
% Type your steady-state relationships.
F(1) = c - y -delta*k;                      % res.constr. (1)
F(2) = 1-beta*(alpha*z*k^(alpha-1)+1-delta); % Euler       (2)
F(3) = y - z*k^alpha;                       % prod. f.    (3)
F(4) = z - 1;                               % AR1         (4)
end
```

Steady-state computation: Method 2

Advanced

- TIP: it's a good idea to simplify as much as you can manually.
  - For example, type this:

$$1 - \beta * (\alpha * z * k^{(\alpha-1)} + 1 - \delta) = 0$$

  - instead of:

$$c^{-\sigma} - \beta * c^{-\sigma} * (\alpha * z * k^{(\alpha-1)} + 1 - \delta) = 0$$

Steady-state computation: Method 2

Advanced

- TIP: it's a good idea to simplify as much as you can manually.
  - For example, type this:

$$1 - \beta * (\alpha * z * k^{(\alpha-1)} + 1 - \delta) = 0$$

  - instead of:

$$c^{-\sigma} - \beta * c^{-\sigma} * (\alpha * z * k^{(\alpha-1)} + 1 - \delta) = 0$$

- Become familiar with `fsolve` options.
  - Tighten tolerance criteria: `FunctionTolerance` and `TolX`.
  - Try different algorithms: `trust-region-dogleg` (default), `trust-region`, and `levenberg-marquardt`.

Steady-state computation: Method 2

Advanced

- You can find the necessary files to implement this method on ILIAS. Go to `Codes`, `Dynare:steady state`.
  - Relevant files are `main2.m` and `growth2.mod`.

Steady-state computation: Method 3

Advanced

- Write a Matlab "steady state function" following Dynare strict rules.
- If the mod file is named `myfile.mod`, then steady state file **must** be called `myfile_steadystate.m`.
- Dynare saves parameter values in structure `M_`.
- In your ss file, you **must** read the parameters from `M_`. You **cannot** pass the parameter values as extra inputs to the function.
- After solving for the ss, you **must** store the results in vector `ys`.
- `ys` **must** be declared both as an input and as an output.
- You **must** include an exit flag, called "check", as a second output.
- You **must** include `exe` as a second input.

Steady-state computation: Method 3

Advanced

- Suppose your mod file is named growth3.mod.

```
function [ys,check] = growth3_steadystate(ys,exe)
global M_
alpha    = M_.params(1);
beta     = M_.params(2);
delta    = M_.params(3);
% Solve for steady-state
% Get c_ss, k_ss, y_ss and z_ss
.......
check = 0;

ys = [log(c_ss);log(k_ss);log(y_ss);log(z_ss)];

end
```

Steady-state computation: Method 3

Advanced

- You can find the necessary files to implement this method on ILIAS. Go to `Codes`, `Dynare:steady state`.
  - Relevant files are `main3.m`, `growth3.mod` and `growth3_steadystate.mod`.

Dynare in matlab programs

- Dynare can be nested into Matlab programs.
- Already seen this in Lecture 3 - looping over parameters.
- But is much more general and useful for:
  - Passing to Dynare steady-state values computed beforehand in Matlab.
  - Nesting mod file inside an estimation routine written in Matlab (GMM, IRF matching, etc.).
- Warning: Dynare clears the memory before starting (so if there are some matlab calculations before, the results will be cancelled).
  - To avoid this use: `dynare myfile.mod noclearall`.

Dynare in matlab programs

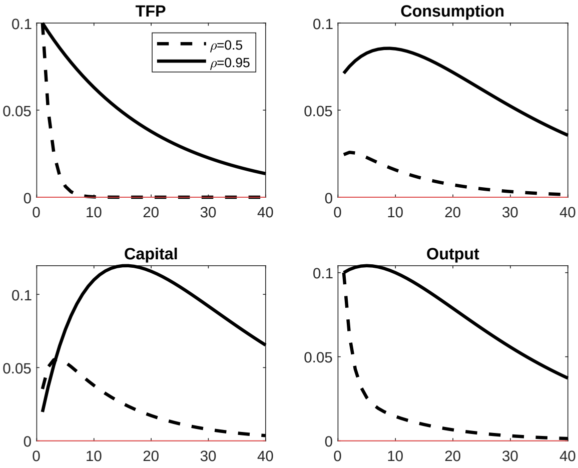
- Suppose you want to produce moments or IRFs for different parameter configurations.
  - How a change in ρ (persistence of TFP shocks) affects the dynamic response of model endogenous variables?
  - How a change in CRRA σ affects the relative standard deviation of consumption w.r.t. output?

Dynare in matlab programs: IRFs

- IRFs of c, k for $\rho \in \{0.5, 0.95\}$

```
rho_vec = [0.5  0.95];
c_irf = zeros(T_irf,2);
k_irf = zeros(T_irf,2);
% Start loop
for i = 1:2
  rho = rho_vec(i);
  save paramfile.mat rho
  dynare growth.mod noclearall
  c_irf(:,i) = c_e;
  k_irf(:,i) = k_e;
end
```

Dynare in matlab programs



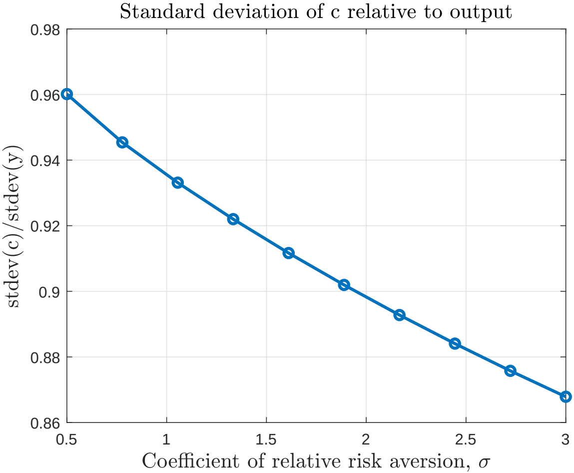
IRF of z, c, k, y to a 0.1 positive shock to z, for $\rho = 0.5$ (dashed line) and $\rho = 0.95$ (solid line).

Dynare in matlab programs: Moments

- Relative stdev. of c for $\sigma \in [0.5, \ldots, 3]$

```
sigma_vec = linspace(0.5,3,np);
c_stdev = zeros(np,1);
% Start loop
for i = 1:np
sigma = sigma_vec(i);
save paramfile.mat sigma
dynare stoch_growth2.mod noclearall
c_stdev(i) = sqrt(oo_.var(c_pos,c_pos))/...
sqrt(oo_.var(y_pos,y_pos));
end
```

Dynare in matlab programs: Moments



Blanchard-Kahn conditions in Dynare

- Recall Blanchard-Kahn conditions seen in Lecture 2:
  - Number of unstable eigenvalues $=$ number of forward-looking variables $\implies$ unique solution.
  - Number of unstable eigenvalues $>$ number of forward-looking variables $\implies$ no solution.
  - Number of unstable eigenvalues $<$ number of forward-looking variables $\implies$ indeterminacy.
- How does this work in Dynare?
- Simple example based on the New-Keynesian model.
  - Dynare error message 1: Example with indeterminacy.
  - Dynare error message 2: Example with no solution.

New-Keynesian model

- Consider plain vanilla NK model:

$$
\begin{aligned}
x_t &= \mathbb{E}_t x_{t+1} - \frac{1}{\sigma}(i_t - \mathbb{E}_t \pi_{t+1} - r) \\
\pi_t &= \beta \mathbb{E}_t \pi_{t+1} + \kappa x_t \\
i_t &= r + \delta \pi_t + v_t \\
v_t &= \rho_v v_{t-1} + \varepsilon_t, \; \varepsilon_t \sim N(0,1).
\end{aligned}
$$

  where x_t is output gap, π_t is inflation rate, i_t is nominal interest rate and v_t is a policy shock.

- Key parameter: δ measures how strongly the central bank reacts to inflation.

New-Keynesian model: Unique Equilibrium?

- The model has a unique stationary solution iff the number of eigenvalues larger than one (in modulus) is equal to the number of forward-looking variables.
- What are the forward-looking variables in this model?

New-Keynesian model: Unique Equilibrium?

- The model has a unique stationary solution iff the number of eigenvalues larger than one (in modulus) is equal to the number of forward-looking variables.
- What are the forward-looking variables in this model?
- Nominal interest rate is static (indeed, it can be eliminated from the system).
- The only predetermined variable is the policy shock v_t (exogenous state variable).

New-Keynesian model: Unique Equilibrium?

- The model has a unique stationary solution iff the number of eigenvalues larger than one (in modulus) is equal to the number of forward-looking variables.
- What are the forward-looking variables in this model?
- Nominal interest rate is static (indeed, it can be eliminated from the system).
- The only predetermined variable is the policy shock v_t (exogenous state variable).
- Inflation π_t and output gap x_t are forward-looking.
- Unique equilibrium requires **two** unstable eigenvalues.

New-Keynesian model

- Write down the model in Dynare:

```
model(linear);
#kappa = (sigma+eta)*((1-omega)*(1-beta*omega)/omega);
// (1) IS curve
x = x(+1) -(1/sigma)*(i-pi(+1)-r);
// (2) NK Philips curve
pi = beta*pi(+1)+kappa*x;
// (3) Taylor rule
i = r+delta*pi+v;
// (4) Policy shock
v = rho_v*v(-1)+e;
end;
```

New-Keynesian model

- Suppose we set $\delta = 0.5$. Any problem?

New-Keynesian model

- Suppose we set $\delta = 0.5$. Any problem?
- Suppose we set $\rho_v = 1.5$. Any problem?

New-Keynesian model

- Suppose we set $\delta = 0.5$. Any problem?
- Suppose we set $\rho_v = 1.5$. Any problem?
- Blanchard-Kahn conditions for unique equilibrium require: $\delta > 1$ and $\rho_v \in (-1, 1)$.
- The restriction $\delta > 1$ is known as the *Taylor principle*.

New-Keynesian model: Indeterminacy

- Set $\delta = 0.5$. Nominal interest rate does not respond strongly enough to inflation $\implies$ multiple equilibria/indeterminacy.
- If you try to run Dynare under such parametrization, you'll get:

```
Error using print_info (line 45)
Blanchard Kahn conditions are not satisfied: indeterminacy

Error in stoch_simul (line 100)
    print_info(info, options_.noprint, options_);

Error in NK (line 142)
info = stoch_simul(var_list_);

Error in dynare (line 235)
evalin('base',fname) ;
```

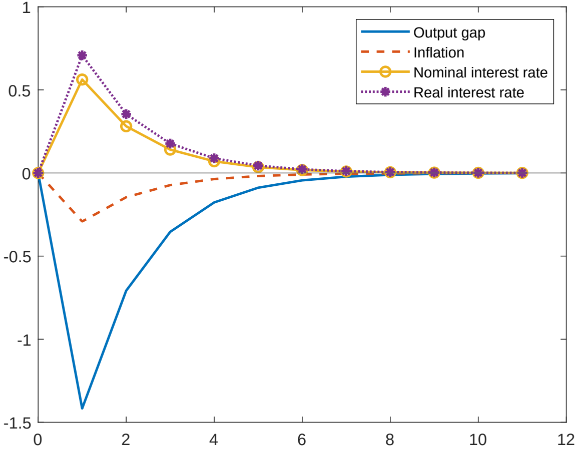
New-Keynesian model: No solution

- Set $\rho_v = 1.5$. The AR(1) process for policy shock is "explosive".
- If you try to run Dynare under such parametrization, you'll get:

```
Error using print_info (line 42)
Blanchard Kahn conditions are not satisfied: no stable equilibrium

Error in stoch_simul (line 100)
    print_info(info, options_.noprint, options_);

Error in NK (line 142)
info = stoch_simul(var_list_);

Error in dynare (line 235)
evalin('base',fname) ;
```

New-Keynesian model: Unique equilibrium

- Set $\delta = 1.5$ and $\rho_v = 0.5$.
- Dynare verifies that BK conditions are satisfied and proceeds with solution.



Correlated shocks

- In simple example with growth model $\implies$ only one exogenous shock ε_t to technology.
- **Assuming** `sig_eps` **denotes the standard deviation**, we can specify the "shock block" in two equivalent ways:

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

```
shocks;
var e; stderr sig_eps;
end;
```

Correlated shocks, cont'd

- Example with two correlated shocks (a_t, z_t):

$$a_t = \rho_a a_{t-1} + \varepsilon_{at},$$
$$z_t = \rho_z z_{t-1} + \varepsilon_{zt}$$

where

$$\begin{bmatrix} \varepsilon_{at} \\ \varepsilon_{zt} \end{bmatrix} \sim N(0, \Sigma).$$

- The variance-covariance matrix is:

$$\Sigma = \begin{bmatrix} \sigma_a^2 & \sigma_{a,z}^2 \\ \sigma_{a,z}^2 & \sigma_z^2 \end{bmatrix}$$

- Note: $COV(\varepsilon_{at}, \varepsilon_{zt}) \equiv \sigma_{a,z}^2 = \phi \cdot \sigma_a \cdot \sigma_z$, where ϕ is the correlation coefficient.

Correlated shocks, cont'd

We have again two equivalent ways to specify the "shocks block":

```
shocks;
var ea = sig_a^2;
var ez = sig_z^2;
var ea,ez = phi*sig_a*sig_z;
end;
```

```
shocks;
var ea; stderr sig_a;
var ez; stderr sig_z;
var ea,ez = phi*sig_a*sig_z;
end;
```

Theoretical vs Simulated Moments

- Suppose we have a solution of a DSGE model.
- How to compute moments (i.e. variance, autocorrelation, etc.) of variables of interest?

Theoretical vs Simulated Moments

- Suppose we have a solution of a DSGE model.
- How to compute moments (i.e. variance, autocorrelation, etc.) of variables of interest?
- Consider simple AR(1) model:

$$y_t = ay_{t-1} + \varepsilon_t \tag{1}$$

  where ε_t is IID shock with mean 0 and variance σ^2.

- How to compute, e.g., $var(y_t) \equiv \gamma_0$?
- One way: simulation
  - Draw a sequence of shocks $\{\varepsilon_t\}_{t=0}^{T}$, with T large.
  - Generate simulated series $\{y_t\}_{t=0}^{T}$ iterating on (1), starting from an arbitrary y_0.

Theoretical vs Simulated Moments

Simulation, cont'd

- Given simulated series $\{y_t\}_{t=0}^{T}$, one can compute variance and autocovariances of y as follows:

$$\hat{\gamma}_0 = \left(\frac{1}{T}\right)\sum_{t=1}^{T}(y_t - \bar{y})^2$$

$$\hat{\gamma}_j = \left(\frac{1}{T}\right)\sum_{t=j+1}^{T}(y_t - \bar{y})(y_{t-j} - \bar{y})^2$$

where $\bar{y}$ is the sample average.

Theoretical vs Simulated Moments

Theoretical Moments

- However, for linear models, the computation of relevant moments can be done also analytically.
- Well known results for (stationary) AR(1) models:

$$Var(y_t) \equiv \gamma_0 = \frac{\sigma^2}{1-a^2}$$
$$Cov(y_t, y_{t-j}) \equiv \gamma_j = a^j \frac{\sigma^2}{1-a^2}$$

for $j = 1, 2, \ldots$

- Of course a law of large numbers will typically ensure that

$$\hat{\gamma}_0 \to \gamma_0, \;\; \hat{\gamma}_j \to \gamma_j \text{ as } T \to \infty$$

Theoretical Moments

How Dynare computes them

- Consider multivariate case $n > 1$.
- DSGE solution can be mapped into a VAR(1):

$$y_t = Ay_{t-1} + \varepsilon_t$$

  where y_t is the vector of endogenous variables and ε_t is vector of exogenous shocks. A is a $n \times n$ matrix and $\varepsilon_t \sim N(0, \Sigma)$ where Σ is the var-cov matrix of ε_t.

- Let $\Gamma_0 = \mathbb{E}(y_t y_t')$ denote the var-cov matrix of y_t and $\Gamma_j = \mathbb{E}\left(y_t y_{t-j}'\right)$ denote the jth order autocovariance matrix.

Theoretical Moments

How Dynare computes them

- (Contemporaneous) variance-covariance matrix of y_t, Γ_0:

$$\begin{aligned}\Gamma_0 &= \mathbb{E}\left(y_t y_t'\right)\\ &= \mathbb{E}\left[\left(Ay_{t-1}+\varepsilon_t\right)\left(Ay_{t-1}+\varepsilon_t\right)'\right]\\ &= A\mathbb{E}\left(y_{t-1}y_{t-1}'\right)A' + \mathbb{E}\left(\varepsilon_t\varepsilon_t'\right)\\ &= A\Gamma_0 A' + \Sigma\end{aligned}$$

- Hence we found that

$$\boxed{\Gamma_0 = A\Gamma_0 A' + \Sigma}$$

- How do we solve for Γ_0?

Theoretical Moments

How Dynare computes them

- Use vec-torization operator to simplify $\Gamma_0 = A\Gamma_0 A' + \Sigma$.
- Recall that

$$vec\,(A+B) = vec(A) + vec(B)$$
$$vec\,(ABC) = \left(C' \otimes A\right) vec\,(B)$$

- it follows that

$$vec\,(\Gamma_0) = vec\left(A\Gamma_0 A'\right) + vec\,(\Sigma)$$
$$vec\,(\Gamma_0) = (A \otimes A)\, vec\,(\Gamma_0) + vec\,(\Sigma)$$

- Hence

$$vec\,(\Gamma_0) = [I - (A \otimes A)]^{-1}\, vec\,(\Sigma)\,.$$

where I has dimension $n^2 \times n^2$. Math

Theoretical Moments

How Dynare computes them

- Likewise, for the autocovariance

$$\begin{aligned}\Gamma_1 &= \mathbb{E}\left(y_t y_{t-1}'\right)\\ &= \mathbb{E}\left[\left(Ay_{t-1}+\varepsilon_t\right)y_{t-1}'\right]\\ &= A\mathbb{E}\left(y_{t-1}y_{t-1}'\right)+\mathbb{E}\left(\varepsilon_t y_{t-1}'\right)\\ &= A\Gamma_0\end{aligned}$$

and in general

$$\begin{aligned}\Gamma_j &= A\Gamma_{j-1}\\ &= A^j\Gamma_0\end{aligned}$$

for $j = 1, 2, \ldots$

Theoretical Moments

How Dynare computes them

- Recall DSGE solution:

$$
\begin{aligned}
s_t &= \Pi s_{t-1} + W\varepsilon_t, \varepsilon_t \sim N(0, \Sigma_\varepsilon) \\
u_t &= U s_t
\end{aligned}
$$

  Clearly, to fit the model into VAR(1) notation, set

$$
\begin{aligned}
A &= \Pi \\
\Sigma &= W \Sigma_\varepsilon W'
\end{aligned}
$$

- In our stochastic growth model Σ_ε is simply a scalar σ_e^2 and $W = [0, 1]'$, so

$$
\Sigma = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \sigma_\varepsilon^2 \begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \sigma_\varepsilon^2 \end{bmatrix}
$$

Theoretical Moments

How Dynare computes them

- Finally, the autocovariance matrix for the control variables in u_t can be computed as:

$$\begin{aligned}\Gamma_j^U &= \mathbb{E}\left(u_t u'_{t-j}\right)\\ &= \mathbb{E}\left[U s_t s'_{t-j} U'\right]\\ &= U\mathbb{E}\left(s_t s'_{t-j}\right)U'\\ &= U\Gamma_j U'.\end{aligned}$$

Simulated Moments

Dynare

- Default option in Dynare: compute theoretical moments.
- If one is interested in empirical moments, set option `periods = INTEGER`.
- If different from zero, Dynare will compute simulation-based moments instead of theoretical moments.
- Dynare will start simulation using the steady-state as initial condition.
- Where are the simulated series for model variables stored?
    - Matrix `oo_.endo_simul`, $n \times T$
- One can also specify `drop = INTEGER` which tells Dynare to discard some initial observations (burn-in).

Appendix

Vec operator

- If A is $m \times n$ matrix, then $vec(A)$ is an $mn \times 1$ column vector, obtained by stacking the columns of A, one below the other.
- For example, if

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}$$

then

$$vec(A) = \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{12} \\ a_{22} \\ a_{32} \end{bmatrix}.$$

Kronecker operator

- Let A be an $m \times n$ matrix and B a $p \times q$ matrix. Then the Kronecker product of A and B is defined as follows:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{bmatrix}$$

  where $A \otimes B$ is an $mp \times nq$ matrix.

- For some useful properties of the Kronecker operator, see Hamilton (1994). Back

Vec and Kronecker operators

Proposition

Let A, B, C be matrices whose dimensions are such that the product ABC exists. Then

$$vec(ABC) = (C' \otimes A)vec(B).$$

Proof.

See Hamilton (1994), Appendix 10.A, page 289. □

Corollary

Let A, B be matrices whose dimensions are such that the product AB exists. Then

$$vec(AB) = (I \otimes A)vec(B).$$