Dynare

Wouter J. Den Haan

University of Amsterdam

July 26, 2010

Introduction

- What is the objective of perturbation?
- Peculiarities of Dynare
- Some examples

Objective of 1st-order perturbation

- Obtain *linear* approximations to the policy functions that satisfy the first-order conditions
- state variables: $x_t = [x_{1,t}\ x_{2,t}\ x_{3,t}\ \cdots x_{n,t}]'$
- result:

$$y_t = \bar{y} + (x_t - \bar{x})'a$$

- a bar above a variable indicates steady state value

Underlying theory

- Model:

$$\mathsf{E}_t\left[f(g(x))\right] = 0,$$

  - $f(x)$ is completely known
  - $g(x)$ is the unknown policy function.
- Perturbation: Solve *sequentially* for the coefficients of the Taylor expansion of $g(x)$.
- More info:
  - notes and slides on perturbation
  - slides on Blanchard-Kahn conditions

Neoclassical growth model

- $x_t = [k_{t-1}, z_t]$
- $y_t = [c_t, k_t, z_t]$
- linearized solution:

$$\begin{aligned} c_t &= \bar{c} + a_{c,k}(k_{t-1} - \bar{k}) + a_{c,z}(z_t - \bar{z}) \\ k_t &= \bar{k} + a_{k,k}(k_{t-1} - \bar{k}) + a_{k,z}(z_t - \bar{z}) \\ z_t &= \rho z_{t-1} + \varepsilon_t \end{aligned}$$

Linear in what variables?

- Dynare does not understand what c_t is.
  - could be level of consumption
  - could be log of consumption
  - could be rainfall in Scotland
- Dynare simply generates a **linear** solution in what you specify as the variables
- More on this below

Peculiarities of Dynare

- Variables known at beginning of period t *must* be dated $t-1$.
- Thus,
  - k_t: the capital stock *chosen* in period t
  - k_{t-1}: the capital stock available at beginning of period t

Peculiarities of Dynare

The solution

$$c_t = \bar{c} + a_{c,k}(k_{t-1} - \bar{k}) + a_{c,z}(z_t - \bar{z})$$
$$k_t = \bar{k} + a_{k,k}(k_{t-1} - \bar{k}) + a_{k,z}(z_t - \bar{z})$$
$$z_t = \rho z_{t-1} + \varepsilon_t$$

can of course be written (less conveniently) as

$$c_t = \bar{c} + a_{c,k}(k_{t-1} - \bar{k}) + a_{c,z_{-1}}(z_{t-1} - \bar{z}) + a_{c,z}\varepsilon_t$$
$$k_t = \bar{k} + a_{k,k}(k_{t-1} - \bar{k}) + a_{k,z_{-1}}(z_{t-1} - \bar{z}) + a_{k,z}\varepsilon_t$$
$$z_t = \rho z_{t-1} + \varepsilon_t$$

with $a_{c,z_{-1}} = \rho a_{c,z}$ and $a_{k,z_{-1}} = \rho a_{k,z}$

Peculiarities of Dynare

- Dynare gives the solution in the less convenient form:

$$c_t = \bar{c} + a_{c,k}(k_{t-1} - \bar{k}) + a_{c,z_{-1}}(z_{t-1} - \bar{z}) + a_{c,z}\varepsilon_t$$
$$k_t = \bar{k} + a_{k,k}(k_{t-1} - \bar{k}) + a_{k,z_{-1}}(z_{t-1} - \bar{z}) + a_{k,z}\varepsilon_t$$
$$z_t = \rho z_{t-1} + \varepsilon_t$$

- Since the Dynare solution satisfies

$$a_{c,z_{-1}} = \rho a_{c,z} \text{ and } a_{k,z_{-1}} = \rho a_{k,z}$$

one could always rewrite the Dynare solution in the more convenient form

Dynare program blocks

- **Labeling block**: indicate which symbols indicate what
  - variables in "var"
  - exogenous shocks in "varexo"
  - parameters in "parameters"
- **Parameter values block**: Assign values to parameters

Dynare program blocks

- **Model block**: Between "model" and "end" write down the n equations for n variables
  - note that dynare has no conditional expectations but if an equation has a $(+1)$ variable, then Dynare knows there is a conditional expectation

Dynare program blocks

- **Initialization block**: Dynare has to solve for the steady state. This can be the most difficult part (since it is a true non-linear problem). So good initial conditions are important
- **Random shock block**: Indicate the standard deviation for the exogenous innovation

Dynare program blocks

- **Solution & Properties block**:
  - Solve the model with the command
    - 1^{st}-order: stoch_simul(order=1,nocorr,nomoments,IRF=0)
    - 2^{nd}-order: stoch_simul(order=2,nocorr,nomoments,IRF=0)
  - Dynare can calculate IRFs and business cycle statistics. E.g.,
    - stoch_simul(order=1,IRF=30),
    - but I would suggest to program this yourself (see below)

Running Dynare

- In Matlab change the directory to the one in which you have your *.mod files
- In the Matlab command window type

dynare programname

- This will create and run several Matlab files

Model with productivity in levels (FOCs A)

Specification of the problem

$$\max_{\{c_t,k_t\}} \mathsf{E} \sum_{t=1}^{\infty} \beta^{t-1} \frac{c_t^{1-\nu}-1}{1-\nu}$$

s.t.

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

$$z_t = (1-\rho) + \rho z_{t-1} + \varepsilon_t$$

k_0 given

$$\mathsf{E}_t[\varepsilon_{t+1}] = 0 \;\&\; \mathsf{E}_t[\varepsilon_{t+1}^2] = \sigma^2$$

Distribution of innovation

- 1^{st}-order approximations:
  - the distribution of ε_t does not matter, except that $\mathsf{E}_t[\varepsilon_{t+1}]$ has to be zero.
- 2^{nd}-order approximations:
  - σ matters (it affects the mean)
  - higher-order moments do not
- Also see notes and slides on perturbation theory

Everything in levels: FOCs A

Model equations:

$$\begin{aligned} c_t^{-\nu} &= \mathsf{E}_t\left[\beta c_{t+1}^{-\nu}(\alpha z_{t+1}k_t^{\alpha-1}+1-\delta)\right] \\ c_t+k_t &= z_t k_{t-1}^{\alpha}+(1-\delta)k_{t-1} \\ z_t &= (1-\rho)+\rho z_{t-1}+\varepsilon_t \end{aligned}$$

Dynare equations:

```
c^(-nu)
 =beta*c(+1)^(-nu)*(alpha*z(+1)*k^(alpha-1)+1-delta);
c+k=z*k(-1)^alpha+(1-delta)k(-1);
z=(1-rho)+rho*z(-1)+e;
```

Policy functions reported by Dynare

- $\delta = 0.025, \nu = 2$, $\alpha = 0.36$, $\beta = 0.99$, and $\rho = 0.95$

POLICY AND TRANSITION FUNCTIONS

	k	z	c
constant	37.989254	1.000000	2.754327
k(-1)	0.976540	-0.000000	0.033561
z(-1)	2.597386	0.950000	0.921470
e	2.734091	1.000000	0.969968

!!!! You have to read output as

	k	z	c
constant	37.989254	1.000000	2.754327
k(-1)-k_{ss}	0.976540	-0.000000	0.033561
z(-1)-z_{ss}	2.597386	0.950000	0.921470
e	2.734091	1.000000	0.969968

- That is, explanatory variables are relative to steady state.
- (Note that steady state of e is zero by definition)
- If explanatory variables take on steady state values, then choices are equal to the constant term, which of course is simply equal to the corresponding steady state value

Changing amount of uncertainty

Suppose $\sigma = 0.1$ instead of 0.007

POLICY AND TRANSITION FUNCTIONS

	k	z	c
constant	37.989254	1.000000	2.754327
k(-1)	0.976540	-0.000000	0.033561
z(-1)	2.597386	0.950000	0.921470
e	2.734091	1.000000	0.969968

- Any change?

Model with productivity in logs

Specification of the problem

$$\max_{\{c_t,k_t\}} \mathsf{E} \sum_{t=1}^{\infty} \beta^{t-1} \frac{c_t^{1-\nu}-1}{1-\nu}$$

s.t.

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

$$z_t = \rho z_{t-1} + \varepsilon_t$$

$$k_0 \text{ given}, \quad \mathsf{E}_t[\varepsilon_{t+1}] = 0$$

Variables in levels & prod. in logs - FOCs B

Model equations:

$$c_t^{-\nu} = \mathsf{E}_t\left[\beta c_{t+1}^{-\nu}(\alpha \exp(z_{t+1})k_t^{\alpha-1} + 1 - \delta)\right]$$
$$c_t + k_t = \exp(z_t)k_{t-1}^{\alpha} + (1-\delta)k_{t-1}$$
$$z_t = \rho z_{t-1} + \varepsilon_t$$

Dynare equations:

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

Policy functions reported by Dynare

- $\delta = 0.025, \nu = 2$, $\alpha = 0.36$ and $\beta = 0.99$

POLICY AND TRANSITION FUNCTIONS

	k	z	c
constant	37.989254	0.000000	2.754327
k(-1)	0.976540	-0.000000	0.033561
z(-1)	2.597386	0.950000	0.921470
e	2.734091	1.000000	0.969968

- What does z stand for here?

Linear solution in what?

Dynare gives a linear system in what you specify the variables to be

All variables in logs - FOCs C

Model equations:

$$(\exp(\tilde{c}_t))^{-\nu} =$$

$$= \mathsf{E}_t \left[\beta(\exp(\tilde{c}_{t+1}))^{-\nu}(\alpha \exp(\tilde{z}_{t+1})(\exp(\tilde{k}_t))^{\alpha-1} + 1 - \delta)\right]$$

$$\exp(\tilde{c}_t) + \exp(\tilde{k}_t) = \exp(\tilde{z}_t)(\exp(\tilde{k}_{t-1}))^{\alpha} + (1-\delta)\exp(\tilde{k}_{t-1})$$

$$\tilde{z}_t = \rho \tilde{z}_{t-1} + \varepsilon_t$$

The variables $\tilde{c}_t$ and $\tilde{k}_t$ are the *log* of consumption and capital.

All variables in logs - FOCs C

Model equations (rewritten a bit)

$$\exp(-\nu\tilde{c}_t) = \mathsf{E}_t\left[\beta\exp(-\nu\tilde{c}_{t+1})(\alpha\exp(\tilde{z}_{t+1}+(\alpha-1)\tilde{k}_t)+1-\delta)\right]$$

$$\begin{aligned}\exp(\tilde{c}_t)+\exp(\tilde{k}_t) &= \exp(\tilde{z}_t+\alpha\tilde{k}_{t-1})+(1-\delta)\exp(\tilde{k}_{t-1})\\ \tilde{z}_t &= \rho\tilde{z}_{t-1}+\varepsilon_t\end{aligned}$$

All variables in logs - FOCs C

Dynare equations:

```
exp(-nu*lc)=beta*exp(-nu*lc(+1))*
(alpha*exp(lz(+1)+(alpha-1)*lk))+1-delta);
exp(lc)+exp(lk)
=exp(lz+alpha*lk(-1))+(1-delta)exp(lk(-1));
lz=rho*lz(-1)+e;
```

All variables in logs - FOCs C

- This system gives policy functions that are linear in the variables `lc`, i.e., $\ln(c_t)$, `lk`,i.e., $\ln(k_t)$, and `lz`, i.e., $\ln(z_t)$,
- Programmers often do not make clear that a variable is a log. That is, they would simply use `c`, `k`, and `z` in the dynare equations above instead of `lc`, `lk`, and `lz`

All variables in logs - FOCs C

Dynare equations (with different notation):

```
exp(-nu*c)=beta*exp(-nu*c(+1))*
(alpha*exp(z(+1)+(alpha-1)*k))+1-delta);
exp(c)+exp(k)=exp(z+alpha*k(-1))+(1-delta)exp(k(-1));
z=rho*z(-1)+e;
```

POLICY AND TRANSITION FUNCTIONS for foc B

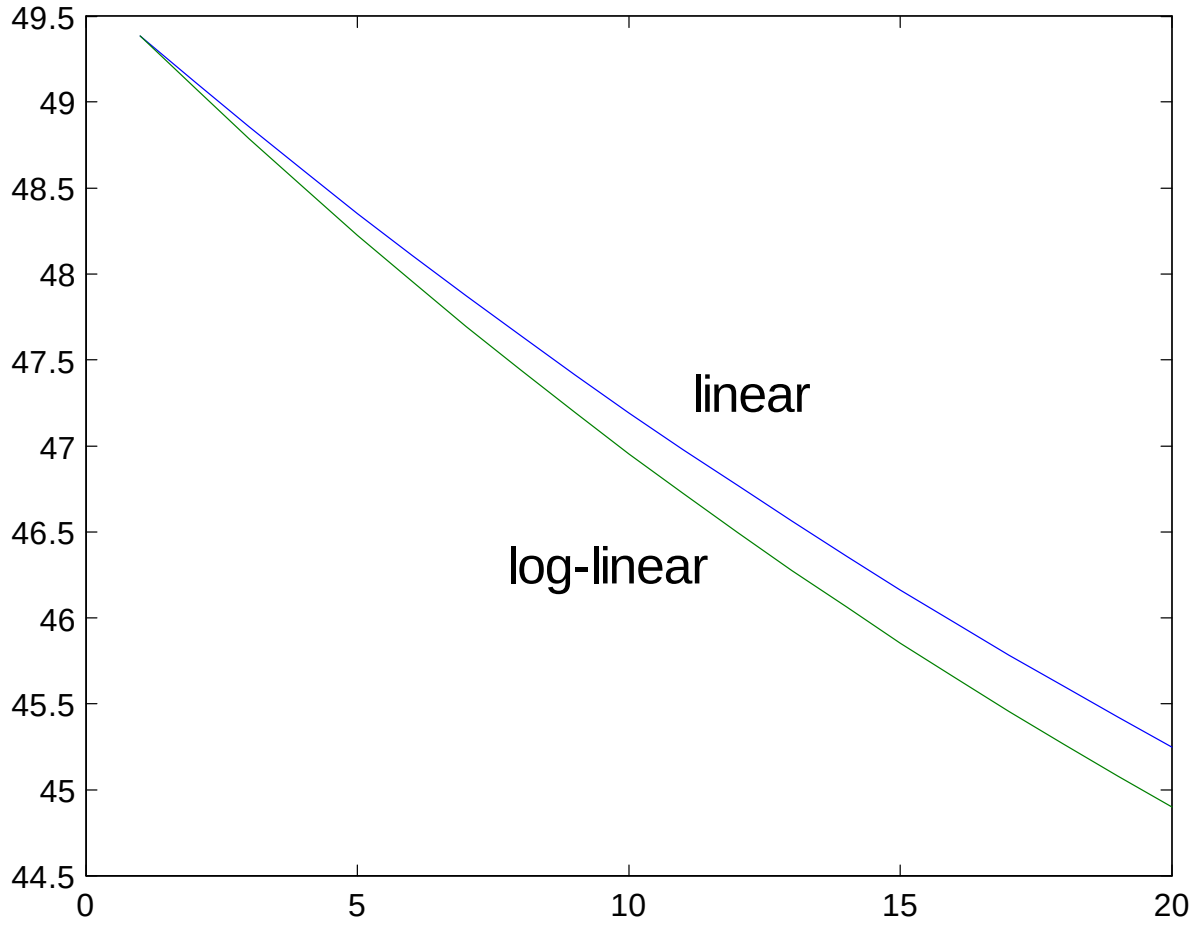
	k	z	c
constant	37.989254	0.000000	2.754327
k(-1)	0.976540	-0.000000	0.033561
z(-1)	2.597386	0.950000	0.921470
e	2.734091	1.000000	0.969968

POLICY AND TRANSITION FUNCTIONS for foc C

	k	z	c
constant	3.637303	0.000000	1.013173
k(-1)	0.976540	0.000000	0.462887
z(-1)	0.068372	0.950000	0.334554
e	0.071970	1.000000	0.352162

These are not the same solutions

Suppose that $k_0 = 49.3860$ & $z_t = 0 \ \forall t$



Example with analytical solution

- If $\delta = \nu = 1$ then we know the analytical solution. It is

$$k_t = \alpha\beta \exp(z_t) k_{t-1}^a$$
$$c_t = (1 - \alpha\beta) \exp(z_t) k_{t-1}^a$$

or

$$\begin{aligned} \ln k_t &= \ln(\alpha\beta) + \alpha \ln k_{t-1} + z_t \\ \ln c_t &= \ln(1 - \alpha\beta) + \alpha \ln k_{t-1} + z_t \end{aligned}$$

- That is, the policy rules are linear in the logs

Dynare solutions I

Dynare equations (with consumption and capital in logs):

```
exp(-c)
=beta*exp(-c(+1))*(alpha*exp(z(+1)+(alpha-1)*k)));
exp(c)+exp(k)=exp(z+alpha*k(-1));
z=rho*z(-1)+e;
```

Dynare solutions I

- Dynare solution ($\alpha = 0.36$ and $\beta = 0.99$)

	k	z	c
constant	-1.612037	0.000000	-1.021009
k(-1)	0.359999	-0.000000	0.360000
z(-1)	0.949998	0.950000	0.950000
e	0.999998	1.000000	1.000000

- Note that c, k, and z are logs
- Check yourself that this is correct

Dynare solutions II

Dynare equations (with consumption and capital in levels):

```
1/c=beta*(1/c(+1))*(alpha*exp(z(+1))*k^(alpha-1));
c+k=exp(z)*k(-1)^alpha;
z=rho*z(-1)+e;
```

Dynare solutions II

- Dynare solution ($\alpha = 0.36$ and $\beta = 0.99$)

	k	z	c
constant	0.199482	0.000000	0.360231
k(-1)	0.360000	0.000000	0.650101
z(-1)	0.189507	0.950000	0.342219
e	0.199482	1.000000	0.360231

- Note that c and k indicate levels and z logs
- This is not the same !!!

Substitute out consumption- FOCs D

Model equations:

$$\left[z_t \exp(\alpha \tilde{k}_{t-1}) + (1-\delta)\exp(\tilde{k}_{t-1}) - \exp(\tilde{k}_t)\right]^{-\nu}$$
$$=$$
$$\mathsf{E}_t\left\{\beta\left(\begin{array}{c}\left[z_{t+1}\exp(\alpha\tilde{k}_t) + (1-\delta)\exp(\tilde{k}_t) - \exp(\tilde{k}_{t+1})\right]^{-\nu} \times \\ (\alpha z_{t+1}\exp((\alpha-1)\tilde{k}_t) + 1 - \delta)\end{array}\right)\right\}$$

$$z_t = (1-\rho) + \rho z_{t-1} + \varepsilon_t$$

Dynare solution

Dynare equations (with capital in logs):

```
(z*exp(alpha*lk(-1)+(1-delta)*exp(lk(-1))-exp(lk))^(-nu)
=beta*(z(+1)*exp(alpha*lk+(1-delta)*exp(lk)-exp(lk(+1))^(
*(alpha*exp(z(+1)+(alpha-1)*lk))+(1-delta));

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

Dynare solution

- Dynare solution ($\alpha = 0.36$ and $\beta = 0.99$)

	lk	z
constant	3.637303	0.000000
lk(-1)	0.976540	0.000000
z(-1)	0.068372	0.950000
e	0.071970	1.000000

- Given this law of motion for $\ln(k_t)$ you can solve c_t using the non-linear equation

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

Do it yourself!

- Try to do as much yourself as possible

What (not) to do your self

- Policy functions:
  - can be quite tricky so let Dynare do it.
- IRFs, business cycle statistics, etc:
  - easy to program yourself
  - you know exactly what you are getting

Why do things yourself?

- Dynare linearizes *everything*
- Suppose you have an RBC in log of capital
  - Add the following equation to introduce investment

$$\exp(i_t) = \exp(k_t) - (1-\delta)\exp(k_{t-1})$$

  - Dynare will approximate this linear equation.

Why do things yourself?

- Now suppose you have an approximation in levels
- Add the following equation to introduce output

$$y_t = z_t k_t^{\alpha} h_t^{1-\alpha}$$

- Dynare will take a first-order condition of this equation to get a first-order approximation for y_t
- But you already have solutions for k_t and h_t

Why do things yourself?

- Getting the policy rules requires a bit of programming
- Thus, it makes sense to use Dynare for this
- But the more you program yourself, the better you understand the results
- Try, therefore, to program the simpler things, like IRFs, simulated time paths, and business cycle statistics yourself, that is, simply use
    - stoch_simul(order=1,nocorr,nomoments,IRF=0)

Tricks

- Incorporating Dynare in other Matlab programs
- Reading parameter values in *.mod file from external file
- Reading Dynare policy functions *as they appear on the screen*
- How to get good initial conditions (to solve for steady state)

Keeping variables in memory

- Dynare clears all variables out of memory
- To overrule this, use

```
dynare program.mod noclearall
```

Saving solution to a file

- Replace the file "disp_dr.m" with the provided file
- I made two changes:
  - The original Dynare file only writes a coefficient to the screen if it exceeds 10^{-6} in absolute value. I eliminated this condition
  - I save the policy functions, *exactly* the way Dynare now writes them to the screen

To load the policy rules into the matrix "decision" simply type

load dynarerocks

Saving solution to a file

- Note that Dynare also saves policy functions, but for second-order this is not what you see on the screen

Saving solution to a file

- Note that Dynare also saves policy functions, but for second-order this is not what you see on the screen

Loops

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

nu=3;

- You would like to run the program twice; once for nu=3, and once for nu=5.

Loops

1. In your Matlab program, loop over the different values of nu. In each iteration, first save the current value of nu (and the associated name) to the file wouterrocks with

   "save parameterfile nu

   and *then* run Dynare
2. In your Dynare program file, replace the command "nu = 3" with

   load parameterfile

   set_param_value('nu',nu);

Using loop to get good initial conditions

With a loop you can update the initial conditions used to solve for steady state

1. Use parameters to definite initial conditions
2. Solve model for simpler case
3. Gradually change parameter
4. You can even gradually change models using weighting coefficients
5. Alternative: (also) use different algorithm to solve for steady state
    1. solve_algo=1,2, or 3
    2. solve for coefficients instead of variables

Simple model with endogenous labor

$$c_t^{-\nu} = \mathsf{E}_t\left[\beta c_{t+1}^{-\nu}\left(\alpha \exp(z_{t+1})\left(k_t/h_{t+1}\right)^{\alpha-1} + 1 - \delta\right)\right]$$

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

$$c_t^{-\nu}(1-\alpha)\exp(z_t)\left(k_{t-1}/h_t\right)^{\alpha} = h_t^{\kappa}$$

$$z_t = \rho z_{t-1} + \varepsilon_t$$

Simple model with endogenous labor

1. Solve for c, k, h using

$$
\begin{aligned}
1 &= \beta(\alpha\,(k/h)^{\alpha-1} + 1 - \delta) \\
c + k &= k^{\alpha} h^{1-\alpha} + (1-\delta)k \\
c^{-\nu}(1-\alpha)(k/h)^{\alpha} &= \phi h^{\kappa} \\
\phi &= 1
\end{aligned}
$$

2. Or solve for c, k, ϕ using

$$
\begin{aligned}
1 &= \beta(\alpha\,(k/h)^{\alpha-1} + 1 - \delta) \\
c + k &= k^{\alpha} h^{1-\alpha} + (1-\delta)k \\
c^{-\nu}(1-\alpha)(k/h)^{\alpha} &= \phi h^{\kappa} \\
h &= 0.3
\end{aligned}
$$

Impulse Response functions

Definition: The effect of a one-standard-deviation shock

- Take as given k_0, z_0, and time series for ε_t, $\{\varepsilon_t\}_{t=1}^{T}$
- Let $\{k_t\}_{t=1}^{T}$ be the corresponding solutions

Impulse Response functions

- Consider the time series ε_t^* such that

$$\begin{aligned} \varepsilon_t^* &= \varepsilon_t \qquad \text{for } t \neq \tau \\ \varepsilon_t^* &= \varepsilon_t + \sigma \text{ for } t = \tau \end{aligned}$$

- Let $\{k_t^*\}_{t=1}^T$ be the corresponding solutions
- Impulse response functions are calculated as

$$IRF_j^k = k_{\tau+j}^* - k_{\tau+j} \quad \text{for } j \geq 0$$

IRFs for linear systems

- Value of $k_{\tau-1}$ and values of original shock $\{\varepsilon_t\}_{t=\tau}^{T}$ irrelevant for IRFs
- Thus, make your life easy by setting
  - $\tau = 1$
  - $k_0 (= k_{\tau-1}) = \bar{k}$
  - $\varepsilon_{\tau+j} = 0$ for $j \geq 0$Take as given k_0, z_0, and time series for ε_t, $\{\varepsilon_t\}_{t=1}^{T}$
- If k is in logs then subtract $\bar{k}$ and you have the IRF
- If k is in levels calculate $(k_{\tau+j} - \bar{k})/\bar{k}$ or $\ln(k_{\tau+j}/\bar{k})$

Impulse Response functions

1st-order case: Dynare gives you

$$k_t = \bar{k} + a_{k,k}(k_{t-1} - \bar{k}) + a_{k,z_{-1}}(z_{t-1} - \bar{z}) + a_{k,\varepsilon}\varepsilon_t$$

- Start at $k_0 = \bar{k}$ and $z_0 = \bar{z}$ $(= 0)$
- Let $\varepsilon_1 = \sigma_\varepsilon$ and $\varepsilon_t = 0$ for $t > 1$
- Calculate time path for z_t
- Calculate time path for k_t
- Calculate time path for other variables
- Calculate % change (subtract steady-state value if variables are in logs)

Impulse Response functions

2nd-order case:

- One could repeat procedure described in last slide
- But with a non-linear law of motion results do depend on initial value of k, realizations of shocks in the original series, and whether $\varepsilon^*_\tau = \varepsilon_\tau + \sigma$ or $\varepsilon^*_\tau = \varepsilon_\tau - \sigma$
  - For example, IRF can be different when initial capital stock is low than when it is high

How to calculate a simulated data set

Dynare gives you

$$k_t = \bar{k} + a_{k,k}(k_{t-1} - \bar{k}) + a_{k,z_{-1}}(z_{t-1} - \bar{z}) + a_{k,\varepsilon}\varepsilon_t$$

- Start at $k_0 = \bar{k}$ and $z_0 = \bar{z}$ $(= 0)$
- Use a random number generator to get a series for ε_t for $t = 1$ to $t = T$
- Calculate time path for z_t
- Calculate time path for k_t
- Calculate time path for other variables
- Discard an initial set of values
- Note that procedure is the same for first and second-order solutions

Simulate higher-order & pruning

- first-order solutions are by construction stationary
  - simulation cannot be problematic
- simulation of higher-order can be problematic
- simulation of 2^{nd}-order will be problematic for large shocks
- trick proposed: **Pruning**
- pruning:
  - is a *trick* to ensure stability
  - it uses a *distorted* numerical approximation

Pruning

- $k^{(n)}(k_{-1}, z)$: the n^{th}-order perturbation solution for k as a function of k_{-1} and z.
- $k_t^{(n)}$: the value of k_t generated with $k^{(n)}(\cdot)$.

Pruning

- For $n > 1$, the regular perturbation solution $k^{(n)}$ can be written as

$$k_t^{(n)} - k_{ss}$$
$$=$$
$$a^{(n)} + a_k^{(n)} \left(k_{t-1}^{(n)} - k_{ss} \right) + a_z^{(n)} \left(z_t - z_{ss} \right)$$
$$+ \tilde{k}^{(n)}(k_{t-1}^{(n)}, z_t)$$

Pruning

- With pruning one would simulate two series

$$k_t^{(1)} - k_{ss} = a_k^{(1)} \left(k_t^{(1)} - k_{ss} \right) + a_z^{(1)} \left(z_t - z_{ss} \right)$$

$$\hat{k}_t^{(n)} - k_{ss} = a^{(n)} + a_k^{(n)} \left(\hat{k}_{t-1}^{(n)} - k_{ss} \right) + a_z^{(n)} \left(z_t - z_{ss} \right) + \tilde{k}^{(n)} \left(k_{t-1}^{(1)}, z_t \right)$$

- $k_t^{(1)}$ is stationary as long as BK conditions are satisfied
- $\tilde{k}^{(n)} \left(k_{t-1}^{(1)}, z_t \right)$ is then also stationary
- $\left| a_1^{(n)} \right| < 1$ then ensures that $\hat{k}_t^{(n)}$ is stationary

Pruning

- The pruned simulated series, $\hat{k}_t^{(n)}$ is *NOT* a function of the corresponding state variables $\hat{k}_t^{(n)}$ and z_t

Practical

- Dynare expects files to be in a regular path like e:\... and cannot deal with subdirectories like //few.eur.nl/.../...
- The solution is to put your *.mod files on a memory stick