INTRODUCTION TO DYNARE

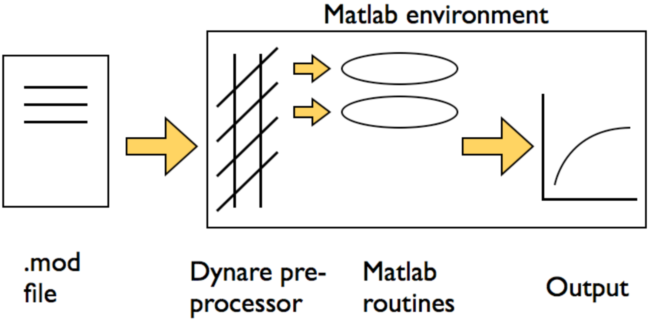
What is Dynare?

- Dynare is a Matlab frontend to solve and simulate dynamic models
- Either deterministic or **stochastic**
- Developed by Michel Juillard at CEPREMAP
- website: `http://www.cepremap.cnrs.fr/dynare/`

How does it work?

- Write the code of the model
- Takes care of parsing the model to Dynare
- Rearrange the model
- Solves the model
- Use the solution to generate some output
- (Can even estimate the model)

How does it Work?



Source: Griffoli (2007)

What will we do?

- Introduction to Dynare
- Very basic linear recursive models (AR(1), Samuelson's Oscillator)
- First linear RE models (very simple cases)
- First linear(ized) "economic model"
- First non–linear model (OGM)
- Fairly general model

How to use Dynare?

Structure of the mod file

Preamble
Define variables and parameters

Model
Equations of the Model

Steady State
Compute the Long–Run

Shocks
Define the properties of Shocks

Solution
Compute the Solution and Produce Output

Structure of the mod file: **Preamble**

- **Aim:** Define variables and parameters
- 3 major instructions:
  1. VAR: Define variables
  2. VAREXO: Define (truely) exogenous variables
  3. PARAMETERS: Declare parameters
- assign values to parameters

Structure of the mod file: **Preamble**

An example

- Assume your model takes the form

$$x_t = \rho x_{t-1} + e_t$$

  with $e_t \rightsquigarrow \mathcal{N}(0, \sigma^2)$
- Variable: x_t
- Exogenous Variable: e_t
- Parameters: ρ, σ

Structure of the mod file: **Preamble**

An example

```
// Simple AR(1) model
// This version: 04/10/08

var x;
varexo e;
parameters rho,sigma;

rho     = 0.90;
sigma   = 0.01;
```

Structure of the mod file: **Model**

- **Aim:** Define model equations
- 1 major instruction:

```
model;

...

end;
```

- write equations as they appear in *natural* language

Structure of the mod file: **Model**

An example

- Again take the AR(1) example: $x_t = \rho x_{t-1} + e_t$
- Model writes:

```
model;

x=rho*x(-1)+e;

end;
```

Structure of the mod file: **Steady State**

- **Aim:** Compute the long–run of the model
- That is: Where its *deterministic* dynamics will converge
- Why? Because it will take a (non–)linear approximation around this long run
- Structure:

```
initval;

...

end;
steady;
check;
```

Structure of the mod file: **Steady State**

- `Steady` computes the long run of the model using a non–linear solver
- Close to the Newton algorithm (more sophisticated though!)
- It therefore needs initial conditions
- That's the role of the `initval;... end;` statement.
- Better give (very) good initial conditions for *all* variables

Structure of the mod file: **Steady State**

- What if you forget `Steady`?
- It will not compute the steady state.
- What happens then?
- Simulations start from values specified in the `initval;... end;` statement.
- `check` is optional. It checks the dynamic stability of the system

Structure of the mod file: **Steady State**

An example

- Again take the AR(1) example: $x_t = \rho x_{t-1} + e_t$
- In deterministic steady state: $e_t = \overline{e} = 0$ therefore

$$\overline{x} = \rho \overline{x} \Longrightarrow \overline{x} = 0$$

- Therefore

```
initval;
e    = 0;
x    = 0;
end;
steady;
check;
```

Structure of the mod file: **Shocks**

- **Aim:** Define the properties of the exogenous shocks
- Exogenous shocks are gaussian innovations.
- They are assumed to be gaussian with $\mathcal{N}(0, \Sigma)$
- Not so limitative actually
- Structure:

```
shocks;
var ...;
stderr ...;

or

var ... = ...;
end;
```

Structure of the mod file: **Shocks**

An example

- Again take the AR(1) example: $x_t = \rho x_{t-1} + e_t$
- Therefore

```
shocks;
var e;
stderr se;
end;
```

or

```
shocks;
var e=se*se;
end;
```

Structure of the mod file: **Solution**

- Final step: Compute the solution and produce some output
- Solution method
  - Deterministic model: Relaxation method
  - Stochastic model: First or Second order perturbation method
- Then compute some moments and impulse responses.
- Getting solution:

```
stoch_simul(...) ...;
```

Structure of the mod file: **Solution**

An example

- Again take the AR(1) example: $x_t = \rho x_{t-1} + e_t$
- Therefore (because the model is linear)

```
stoch_simul(linear);
```

Structure of the mod file: **Solution**

Options of the `stoch_simul` option

- Solver
  - `linear`: In case of a linear model.
  - `order` = 1 or 2 : order of Taylor approximation (default = 2).
- Output (prints everything by default)
  - `noprint`: cancel any printing.
  - `nocorr`: doesn't print the correlation matrix.
  - `nofunctions`: doesn't print the approximated solution.
  - `nomoments`: doesn't print moments of the endogenous variables.
  - `ar` = INTEGER: Order of autocorrelation coefficients to compute (5)

Structure of the mod file: **Solution**

Options of the `stoch_simul` option

- Impulse Response Functions
  - `irf` = INTEGER: number of periods on which to compute the IRFs (Setting IRF=0, suppresses the plotting of IRFs.
  - `relative_irf` requests the computation of normalized IRFs in percentage of the standard error of each shock.
- Simulations
  - `periods` = INTEGER: specifies the number of periods to use in simulations (default = 0).
  - `replic` = INTEGER: number of simulated series used to compute the IRFs (default = 1 if order = 1, and 50 otherwise).
  - `drop` = INTEGER: number of points dropped in simulations (default = 100).
  - `simul_seed` = INTEGER or DOUBLE or (EXPRESSION): specifies a seed for the random number generator.

Complete mod file (ar1.mod)

```
//
// AR(1) model
//
var x;                // Name of the variable
varexo e;             // Name of the exogenous variable
parameters rho se;    // Parameters of the model

rho = 0.95;
se  = 0.02;

model;
x   = rho*x(-1)+e;
end;

initval;
e=0;
x=0;
end;
steady;
check;

shocks;
var e; stderr se;
end;

stoch_simul(linear);
```

Invoking Dynare

```
dynare ar1
```

Typical Output: ar1.log

```
STEADY-STATE RESULTS:
x         0

EIGENVALUES:
         Modulus              Real        Imaginary
            0.95              0.95                0

There are 0 eigenvalue(s) larger than 1 in modulus
for 0 forward-looking variable(s)

The rank condition is verified.

MODEL SUMMARY

  Number of variables:          1
  Number of stochastic shocks: 1
  Number of state variables:    1
  Number of jumpers:            0
  Number of static variables:   0
```

Typical Output: ar1.log

```
MATRIX OF COVARIANCE OF EXOGENOUS SHOCKS

Variables     e
e             0.000400

POLICY AND TRANSITION FUNCTIONS
                                      x
x(-1)                          0.950000
e                              1.000000

THEORETICAL MOMENTS
VARIABLE     MEAN        STD. DEV.  VARIANCE
x               0.0000      0.0641      0.0041

MATRIX OF CORRELATIONS
Variables     x
x             1.0000

COEFFICIENTS OF AUTOCORRELATION
Order     1       2       3       4       5
x         0.9500  0.9025  0.8574  0.8145  0.7738
```

Learning Dynare

- Best thing to do to learn dynare
- Practice Dynare!
- We will now go from simple to more and more complex models

Learning Dynare: Backward looking models

Samuelson's Oscillator

- Samuelson's accelerator

$$
\begin{aligned}
C_t &= cY_{t-1} \\
I_t &= b(C_t - C_{t-1}) \\
G_t &= \rho G_{t-1} + (1-\rho)\overline{G} + \varepsilon_t \\
Y_t &= C_t + I_t + G_t
\end{aligned}
$$

- Variables: C_t, I_t, Y_t, G_t
- Exogenous Variable: ε_t
- Parameters: b, c, ρ, σ

Learning Dynare: Backward looking models

Samuelson's Oscillator (samuelson.mod)

```
//
// Samuelson's Oscillator
//
var ct,it,yt,gt;            // Name of the variable
varexo eg;                  // Name of the exogenous variable
parameters b,c,rhog,gb,sg;  // Parameters of the model

c   = 0.8;
b   = 1.05;
rhog= 0.9;
sg  = 0.02;
gb  = 1;

// Equations of the model

model;
ct  = c*yt(-1);
it  = b*(ct-ct(-1));
yt  = ct+it+gt;
gt  = rhog*gt(-1)+(1-rhog)*gb+eg;
end;
```

Learning Dynare: Backward looking models

Samuelson's Oscillator (samuelson.mod)

```
initval;
eg  = 0;
gt  = gb;
it  = 0;
yt  = gb/(1-c);
ct  = c*yt;
end;
steady;
check;

// Declaring the shocks

shocks;
var eg; stderr sg;
end;

// Launch solving procedure

stoch_simul(linear,irf=100);
```

Learning Dynare: Backward looking models

- Always the same structure
- Only constraint: make sure the model is stable!
- For instance, let's run the oscillator with `b=2`;

Learning Dynare: Forward looking models

- Our first linear rational expectations model
- Postpone theory to tomorrow!
- The model writes

$$y_t = aE_t y_{t+1} + bx_t$$
$$x_t = \rho x_{t-1} + \varepsilon_t$$

- Variables: x_t, y_t
- Exogenous Variable: ε_t
- Parameters: a, b, ρ, σ

Learning Dynare: Forward looking models

- ε_t is exogenous
- x_t is a **predetermined** variable
- y_t is a jump variable

$$y_t = \sum_{j=0}^{\infty} a^j E_t x_{t+j}$$

- Blanchard and Kahn (Econometrica 1980)
- Fundamentally forward looking model!
- Dynare knows how to solve it!

Learning Dynare: Forward looking models

Linear RE model (`linre.mod`)

```
//
// Basic linear RE model
//
var y,x;                // Name of the variable
varexo e;               // Name of the exogenous variable
parameters a,b,rho,se;  // Parameters of the model

rho = 0.95;
se  = 0.02;
a   = 0.8;
b   = 1;

// Equations of the model

model;
y   = a*y(+1)+b*x;
x   = rho*x(-1)+e;
end;
```

Learning Dynare: Forward looking models

Linear RE model (linre.mod)

```
initval;
e=0;
x=0;
y=0;
end;
steady;

// Checking Dynamic Properties

check;

// Declaring the shocks

shocks;
var e; stderr se;
end;

// Launch solving procedure

stoch_simul(linear,irf=20,relative_irf);
```

Learning Dynare: Backward–Forward looking models

- Mix of jump and predetermined (endogenous) variables
- The model writes

$$E_t y_{t+1} - (\lambda + \mu) y_t + \lambda\mu y_{t-1} = b x_t$$
$$x_t = \rho x_{t-1} + \varepsilon_t$$

- Variables: x_t, y_t
- Exogenous Variable: ε_t
- Parameters: $\lambda, \mu, b, \rho, \sigma$

Learning Dynare: Backward–Forward looking models

- ε_t is exogenous
- x_t is a **predetermined** variable
- y_t is a jump variable but it has also a predetermined component!!!!!
- Blanchard and Kahn (Econometrica 1980) again!

Learning Dynare: Backward–Forward looking models

Linear RE model (linrebf.mod)

```
//
// Basic linear RE model
//
var y,x;                      // Name of the variable
varexo e;                     // Name of the exogenous variable
parameters lb,mu,b,rho,se;    // Parameters of the model

rho = 0.95;
se  = 0.02;
mu  = 0.8;
lb  = 1.2;
b   =-1;

// Equations of the model

model;
y(+1)-(lb+mu)*y+lb*mu*y(-1)=b*x;
x   = rho*x(-1)+e;
end;
```

Learning Dynare: Backward–Forward looking models

Linear RE model (linrebf.mod)

```
initval;
e=0;
x=0;
y=0;
end;
steady;

// Checking Dynamic Properties

check;

// Declaring the shocks

shocks;
var e; stderr se;
end;

// Launch solving procedure

stoch_simul(linear,irf=20,relative_irf);
```

Learning Dynare: An new dynamic IS–LM model

- Basic new Keynesian model
- Dynamic IS curve + Phillips curve + Monetary Policy
- See notes on website (beyond the scope of these lectures)

Learning Dynare: An new dynamic IS–LM model

Spelling out the model

$$
\begin{aligned}
IS:&\ \frac{\mathbb{E}_t\widehat{y}_{t+1}}{1-b} - \frac{1+b}{1-b}\widehat{y}_t + \frac{b}{1-b}\widehat{y}_{t-1} + \mathbb{E}_t\widehat{\pi}_{t+1} = \widehat{R}_t + \mathbb{E}_t\widehat{\nu}_{t+1} - \widehat{\nu}_t \\
PC:&\ \widehat{\pi}_t = \frac{(1-\gamma)(1+\beta\gamma)}{(1-\gamma)(1+\beta\zeta)}\widehat{x}_t + \frac{\beta}{1+\beta\zeta}\mathbb{E}_t\widehat{\pi}_{t+1} + \frac{\zeta}{1+\beta\zeta}\widehat{\pi}_{t-1} \\
TR:&\ \widehat{R}_t = \rho_r\widehat{R}_{t-1} + (1-\rho_r)\left(\alpha_\pi\widehat{\pi}_t + \alpha_y\widehat{y}_t\right) + \varepsilon_t^r \\
x:&\ x_t = \frac{1+\sigma_h(1-b)}{1-b}\widehat{y}_t - \frac{b}{1-b}\widehat{y}_{t-1} - (1+\sigma_h)\widehat{a}_t \\
a:&\ \widehat{a}_t = \rho_a\widehat{a}_{t-1} + \varepsilon_t^a \\
\nu:&\ \widehat{\nu}_t = \rho_\nu\widehat{\nu}_{t-1} + \varepsilon_t^\nu
\end{aligned}
$$

Learning Dynare: An new dynamic IS–LM model

- Mix of jump and predetermined (endogenous) variables
- (Truly) Endogenous variables: $\widehat{y}_t$, $\widehat{\pi}_t$, $\widehat{R}_t$, $(\widehat{x}_t)$
- All have both a forward and a backward looking component
- Forcing variables: $\widehat{a}_t$, $\widehat{\nu}_t$
- Truly exogenous variables: ε_t^a, ε_t^ν, ε_t^r.
- Variables: $\widehat{y}_t$, $\widehat{\pi}_t$, $\widehat{R}_t$, $\widehat{x}_t$,$\widehat{a}_t$, $\widehat{\nu}_t$.
- Exogenous Variable: ε_t^a, ε_t^ν, ε_t^r
- Parameters: β, b, σ_h, γ, ζ, ρ_r, α_π, α_y, ρ_a, ρ_ν, σ_a, σ_ν, σ_r.

Learning Dynare: An new dynamic IS–LM model

dynasad.mod

```
var y,dp,x,r,a,u;       // Name of the variable
varexo ea,eu,er;        // Name of the exogenous variable
parameters  b,beta,sh,  // Preferences
            gam,zeta,   // Calvo contracts
            rr,ap,ay,sr // Taylor rule
            ra,sa,      // Supply Shock
            ru,su;      // Preference Shock

// Assigning values
b   = 0.8;      // Habit preferences
beta= 0.99;     // Discount factor
sh  = 1;        // Inverse of labor supply elasticity
gam = 0.75;     // Probability of keeping price
zeta= 1;        // Indexation
rr  = 0.75;     // Interest rate smoothing
ap  = 1.5;      // Reaction to Inflation
ay  = 0.15;     // Reaction to output
sr  = 0.01;     // Std. dev. of Ponetary Policy Shock
ra  = 0.95;     // Persistence of Technology Shock
sa  = 0.01;     // Std. dev. of Technology Shock
ru  = 0.95;     // Persistence of preference shock
su  = 0.01;     // Std. dev. of preference shock
```

Learning Dynare: An new dynamic IS–LM model

dynasad.mod

```
model;
// Definition of markup
x   = (1+sh*(1-b))*y/(1-b)-b*y(-1)/(1-b)-(1+sh)*a;

// IS curve
y(+1)-(1+b)*y+b*y(-1)+(1-b)*dp(+1)=(1+b)*(R+u(+1)-u);

// Phillips curve
dp  = (1-gam)*(1-beta*gam)*x/((1+beta*zeta)*(1-gam))+beta*dp(+1)/(1+beta*zeta)+zeta*dp(-1)/(1+beta*zeta);

// Taylor Rule
r   = rr*r(-1)+(1-rr)*(ap*dp+ay*y)+er;

// Supply Shock
a   = ra*a(-1)+ea;

// Preference Shock
u   = ru*u(-1)+eu;
end;
```

Learning Dynare: An new dynamic IS–LM model

dynasad.mod

```
// Solving the Steady State

initval;
ea  = 0;
er  = 0;
eu  = 0;
a   = 0;
u   = 0;
y   = 0;
x   = 0;
r   = 0;
dp  = 0;
end;
steady;
check;

// Declaring the shocks

shocks;
var ea; stderr sa;
var er; stderr sr;
var eu; stderr su;
end;

// Launch solving procedure

stoch_simul(linear,irf=20) y,dp,t;
```

Learning Dynare: An new dynamic IS–LM model

Output

```
STEADY-STATE RESULTS:
a        0
dp       0
r        0
u        0
x        0
y        0

EIGENVALUES:
         Modulus             Real          Imaginary
          0.4493           0.3312             0.3036
          0.4493           0.3312            -0.3036
          0.6742           0.6742                  0
            0.95             0.95                  0
            0.95             0.95                  0
            2.11            1.802              1.099
            2.11            1.802             -1.099
             Inf              Inf                  0


There are 3 eigenvalue(s) larger than 1 in modulus
for 3 forward-looking variable(s)
The rank condition is verified.
```

Learning Dynare: An new dynamic IS–LM model

Output

```
MODEL SUMMARY
  Number of variables:         6
  Number of stochastic shocks: 3
  Number of state variables:   5
  Number of jumpers:           3
  Number of static variables:  1

MATRIX OF COVARIANCE OF EXOGENOUS SHOCKS
Variables     ea       er       eu
ea            0.000100 0.000000 0.000000
er            0.000000 0.000100 0.000000
eu            0.000000 0.000000 0.000100

POLICY AND TRANSITION FUNCTIONS
                                y           dp            r
a (-1)                   0.272562    -0.075061    -0.017927
r (-1)                  -0.763472    -0.880148     0.391314
dp(-1)                  -0.284859     0.282078     0.095097
u (-1)                   0.049028     0.066723     0.026860
y (-1)                   0.663157    -0.019617     0.017512
ea                       0.286907    -0.079011    -0.018870
er                      -1.017962    -1.173530     0.521753
eu                       0.051609     0.070235     0.028273
```

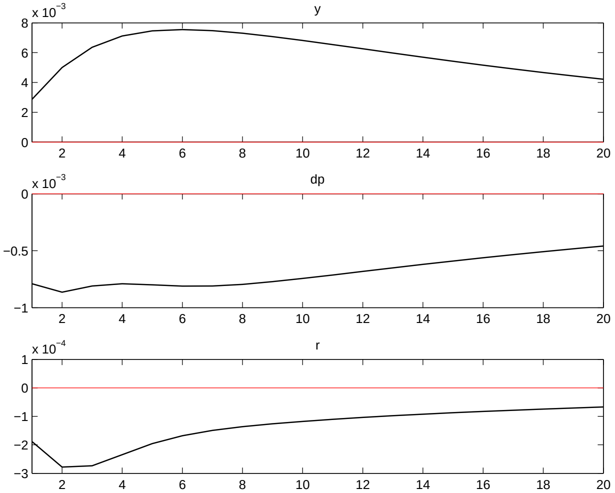
Learning Dynare: An new dynamic IS–LM model

Output

```
THEORETICAL MOMENTS

VARIABLE     MEAN        STD. DEV.   VARIANCE
y               0.0000      0.0327      0.0011
dp              0.0000      0.0148      0.0002
r               0.0000      0.0056      0.0000

VARIANCE DECOMPOSITION (in percent)
         ea      eu      er
y      84.02    0.05   15.93
dp      5.40    0.77   93.83
r       1.58    8.53   89.89

MATRIX OF CORRELATIONS
Variables     y        dp       r
y            1.0000  0.1756 -0.4074
dp           0.1756  1.0000 -0.7379
r           -0.4074 -0.7379  1.0000

COEFFICIENTS OF AUTOCORRELATION
Order    1       2       3       4       5
y        0.9292  0.8526  0.7993  0.7617  0.7295
dp       0.5712  0.2018  0.0441  0.0141  0.0255
r        0.2224 -0.0080 -0.0058  0.0402  0.0682
```

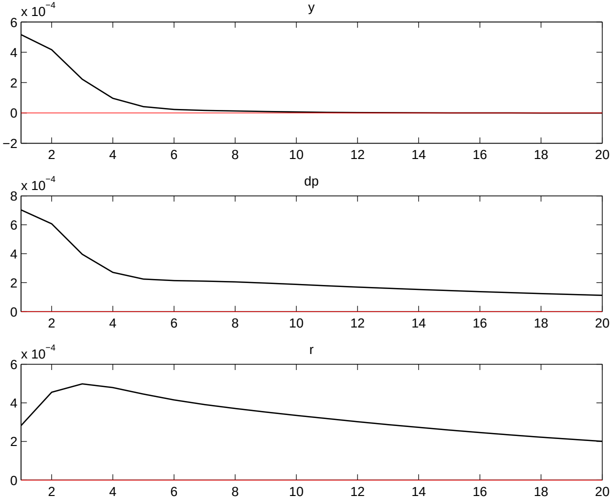
Learning Dynare: An new dynamic IS–LM model

Output: IRF to Supply Shock



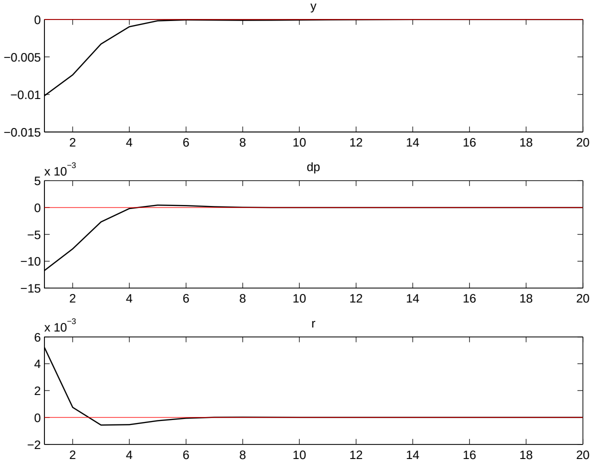
Learning Dynare: An new dynamic IS–LM model

Output: IRF to Preference Shock



Learning Dynare: An new dynamic IS–LM model

Output: IRF to Monetary Policy Shock



- So far all the models we considered were linear
- Most economic models are not!
- Dynare deals with this!
- Dynare actually linearizes (or take a second order Taylor expansion) the model around the steady state
- Let's use an RBC model to illustrate this
- In fact we can even take a log–linear approximation

The RBC Model

- The problem

$$\max \mathbb{E}_t \sum_{\tau=t}^{\infty} \beta^{\tau-t} \left[\log(c_\tau) - \exp(\nu_\tau) \frac{h_\tau^{1+\sigma_h}}{1+\sigma_h} \right]$$

Subject to

$$\begin{aligned} k_{t+1} &= \exp(a_t) k_t^{\alpha} h_t^{1-\alpha} - c_t + (1-\delta) k_t \\ a_t &= \rho_a a_{t-1} + \varepsilon_t^a \text{ and } \nu_t = \rho_\nu \nu_{t-1} + (1-\rho_\nu)\overline{\nu} + \varepsilon_t^{\nu} \end{aligned}$$

- First order conditions:

$$\begin{aligned} c_t^{-1} &= \lambda_t \\ \nu_t h_t^{\sigma_h} &= \lambda_t (1-\alpha) \frac{y_t}{h_t} \\ \lambda_t &= \beta \mathbb{E}_t \lambda_{t+1} \left(\alpha \frac{y_{t+1}}{k_{t+1}} + 1 - \delta \right) \end{aligned}$$

The RBC Model

Collecting equations

$$
\begin{aligned}
y_t &= \exp(a_t) k_t^{\alpha} h_t^{1-\alpha} \\
y_t &= c_t + i_t \\
c_t^{-1} &= \lambda_t \\
\exp(\nu_t) h_t^{\sigma_h} &= \lambda_t (1-\alpha) \frac{y_t}{h_t} \\
\lambda_t &= \beta \mathbb{E}_t \lambda_{t+1} \left(\alpha \frac{y_{t+1}}{k_{t+1}} + 1 - \delta \right) \\
k_{t+1} &= i_t + (1-\delta) k_t \\
a_t &= \rho_a a_{t-1} + \varepsilon_t^a \\
\nu_t &= \rho_\nu \nu_{t-1} + (1-\rho_\nu)\overline{\nu} + \varepsilon_t^\nu
\end{aligned}
$$

The RBC Model

- Mix of jump and predetermined (endogenous) variables
- Endogenous variables: y_t, c_t, i_t, k_t, h_t, λ_t.
- Forcing variables: $\widehat{a}_t$, $\widehat{\nu}_t$.
- Truly exogenous variables: ε_t^a, ε_t^ν.
- Variables: y_t, c_t, i_t, k_t, h_t, λ_t, $\widehat{a}_t$, $\widehat{\nu}_t$.
- Exogenous Variable: ε_t^a, ε_t^ν
- Parameters: β, σ_h, α, δ, $\overline{\nu}$, ρ_a, ρ_ν, σ_a, σ_ν.

The RBC Model

rbc.mod

```
var y,c,i,k,h,lb,a,nu;  // Name of the variable
varexo ea,eu;           // Name of the exogenous variable
parameters  beta,sh,    // Preferences
            alpha,delta,// Technology
            ra,sa,      // Technology Shock
            ru,su,nub;  // Preference Shock
beta    = 0.99;         // Discount factor
sh      = 1;            // Inverse of labor supply elasticity
alpha   = 0.35;         // capital elasticity
delta   = 0.025;        // depreciation rate
ra      = 0.95;         // Persistence of Technology Shock
sa      = 0.01;         // Std. dev. of Technology Shock
ru      = 0.95;         // Persistence of preference shock
su      = 0.01;         // Std. dev. of preference shock

// Assisting steady state
ysk     = (1-beta*(1-delta))/(alpha*beta);
ksy     = 1/ysk;
isy     = delta/ysk;
csy     = 1-isy;
hs      = 0.3;
ys      = ksy^(alpha/(1-alpha))*hs;
cs      = csy*ys;
is      = isy*ys;
ks      = ksy*ys;
lbs     = 1/cs;
nub     =(hs^(1+sh))/((1-alpha)*ys*lbs);
```

The RBC Model

rbc.mod

```
// Equations of the model
// We will use log-linear representation: Y = exp(y)
model;
y=a+alpha*k(-1)+(1-alpha)*h;                          // Production function

exp(y)=exp(c)+exp(i);                                 // Equilibrium

-c=lb;                                                // Consumption

-nu+(1+sh)*h=log(1-alpha)+lb+y;                       // Labor market

exp(k)=exp(i)+(1-delta)*exp(k(-1));                   // Capital accumulation

exp(lb)=beta*exp(lb(+1))*(alpha*exp(y(+1)-k)+1-delta); // Euler equation

a   = ra*a(-1)+ea;                                    // Supply Shock

nu   = ru*nu(-1)+(1-ru)*log(nub)+eu;                  // Preference Shock
end;
```

The RBC Model

rbc.mod

```
initval;
ea  = 0;
eu  = 0;
a   = 0;
nu  = log(nub);
y   = log(ys);
c   = log(cs);
i   = log(is);
k   = log(ks);
h   = log(hs);
lb  = log(lbs);
end;
steady;
check;

// Declaring the shocks
shocks;
var ea; stderr sa;
var eu; stderr su;
end;

// Launch solving procedure
stoch_simul(linear,irf=20,hp_filter=1600) y,c,i,k,h;
```

The RBC Model

Output

```
STEADY-STATE RESULTS:
a        0
c        -0.252394
h        -1.20397
i        -1.35485
k        2.33403
lb       0.252394
nu       -2.26389
y        0.0343289

EIGENVALUES:
         Modulus             Real        Imaginary
            0.95             0.95                0
            0.95             0.95                0
          0.9561           0.9561                0
           1.056            1.056                0
             Inf              Inf                0
There are 2 eigenvalue(s) larger than 1 in modulus for 2 forward-looking variable(s)
The rank condition is verified.

MODEL SUMMARY
  Number of variables:          8
  Number of stochastic shocks: 2
  Number of state variables:   3
  Number of jumpers:            2
  Number of static variables:  3
```

The RBC Model

Output

MATRIX OF COVARIANCE OF EXOGENOUS SHOCKS

Variables	ea	eu
ea	0.000100	0.000000
eu	0.000000	0.000100

POLICY AND TRANSITION FUNCTIONS

	y	c	i	k	h
Constant	0.034329	-0.252394	-1.354847	2.334032	-1.203973
a (-1)	1.237402	0.353088	3.900553	0.097514	0.442157
k (-1)	0.242327	0.573628	-0.755400	0.956115	-0.165650
nu(-1)	0.402156	0.114754	1.267680	0.031692	0.618701
ea	1.302529	0.371672	4.105845	0.102646	0.465429
eu	0.423322	0.120793	1.334400	0.033360	0.651264

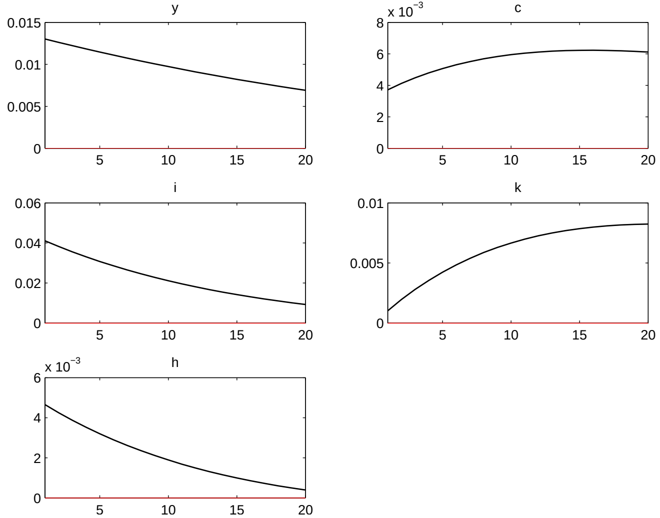
The RBC Model

Output

```
THEORETICAL MOMENTS (HP filter, lambda = 1600)

VARIABLE    MEAN       STD. DEV.  VARIANCE
y             0.0343     0.0179     0.0003
c            -0.2524     0.0059     0.0000
i            -1.3548     0.0564     0.0032
k             2.3340     0.0050     0.0000
h            -1.2040     0.0105     0.0001

MATRIX OF CORRELATIONS (HP filter, lambda = 1600)
Variables     y       c       i       k       h
y             1.0000  0.9045  0.9910  0.3569  0.7945
c             0.9045  1.0000  0.8393  0.7213  0.6620
i             0.9910  0.8393  1.0000  0.2288  0.8051
k             0.3569  0.7213  0.2288  1.0000  0.1597
h             0.7945  0.6620  0.8051  0.1597  1.0000

COEFFICIENTS OF AUTOCORRELATION (HP filter, lambda = 1600)
Order     1       2       3       4       5
y         0.7196  0.4812  0.2828  0.1217 -0.0055
c         0.8045  0.6156  0.4392  0.2794  0.1387
i         0.7092  0.4646  0.2635  0.1022 -0.0233
k         0.9592  0.8607  0.7245  0.5671  0.4017
h         0.7099  0.4657  0.2648  0.1035 -0.0221
```

The RBC Model

IRF to Technology Shock



The RBC Model

IRF to Preference Shock