Solving the RBC Model in Dynare
Economics 210C

Johannes Wieland
jfwieland@ucsd.edu

Spring 2021

RBC Model

- Nonlinear equilibrium conditions:

$$
\begin{aligned}
(1-\alpha)\frac{Y_t}{N_t} &= \frac{\chi N_t^{\varphi}}{C_t^{-\gamma}} \\
1 &= E_t\left\{\frac{\beta C_{t+1}^{-\gamma}}{C_t^{-\gamma}}\left[\alpha\frac{Y_{t+1}}{K_t}+(1-\delta)\right]\right\} \\
Y_t &= C_t + K_t - (1-\delta)K_{t-1} \\
Y_t &= A_t K_{t-1}^{\alpha} N_t^{1-\alpha} \\
A_t &= A_{t-1}^{\rho_a} e^{\varepsilon_t^a}
\end{aligned}
$$

Why Dynare?

- Can solve the log-linearized model by hand.
    - Useful to gain intuition.
- But:
    - Easy to make mistakes.
    - Is the solution unique?
    - Gets harder for more complex models.
    - How to estimate the model?

DYNARE

- Collection of very powerful Matlab codes:
  - Log-linearize, find steady state, compute the model, estimate parameters...
  - Stable, efficient, well-tested codes.
- Available at https://www.dynare.org/download/.

Writing the Dynare Code

- Dynare code is written in a ".mod" file.
- We will create a file called "RBC.mod."
- First step is to create variables of the model.
    - I will use lower case letters to denote log-deviations.
    - So the variables in our model are y, n, c, k, a.

RBC.MOD FILE

- "var" tells Dynare that a list of variables follows.

- Every line in Dynare ends with a semi-colon.

```
%--------------------------------------------------------
% 1. Defining variables
%--------------------------------------------------------

var y, n, c, k, a;
```

RBC.MOD FILE

- "varexo" declares the shocks.
- Note: this just includes the shocks. The TFP process (A_t) is included with the other endogenous variables.

```
%----------------------------------------------------------------
% 1. Defining variables
%----------------------------------------------------------------

var y, n, c, k, a;

varexo e_a;
```

RBC.MOD FILE

- Next we declare the parameters of the model using "parameters".
- Good habit to write out the names and use subscripts for the shock parameters.

```
%----------------------------------------------------------------
% 1. Defining variables
%----------------------------------------------------------------

var y, n, c, k, a;

varexo e_a;

parameters alpha, beta, gamma, delta, chi, phi, rho_a, sigma_a
```

Parameter Values

```
%-------------------------------------------------------
% 2. Calibration
%-------------------------------------------------------

alpha  = 1/3;
beta  = 0.984;
gamma = 1;
chi = 3.48;
phi  = 1/4;
delta = 0.025;
rho_a = 0.979;
sigma_a = 0.0072;
```

Declaring the Model

- In this step we tell Dynare the first order conditions of our model.
- This is done inside the Model block, which begins with "model;" and ends with "end;".
- We want Dynare to log-linearize the model for us.
  - That is why we declared the lower-case letters as variables.
- So rather than write A_t in the model file, we write $exp(a_t)$.

A Static FOC

- Take our FOC for labor supply and labor demand:

$$(1-\alpha)\frac{Y_t}{N_t} = \frac{\chi N_t^{\varphi}}{C_t^{-\gamma}}$$

- This becomes:

```
(1-alpha) * exp(y) / exp(n) = chi * exp(n)^phi * exp(c)^gamma;
```

Dating Variables

- Previous example was special in that all variables are dated "t".
- If there is an intertemporal dimension, we tell Dynare as follows:
  - x_{t+1} is $x(+1)$.
  - x_{t-1} is $x(-1)$.
- E.g., the production function:

$$Y_t = A_t K_{t-1}^{\alpha} N_t^{1-\alpha}$$

```
exp(y) = exp(a) * exp(k(-1))^alpha * exp(n)^(1-alpha);
```

The Model Block

```
%----------------------------------------------------------
% 3. Model
%----------------------------------------------------------
model;

(1-alpha) * exp(y) / exp(n) = chi * exp(n)^phi * exp(c)^gamma;

exp(c)^(-gamma) = beta * exp(c(+1))^(-gamma)
*(  alpha * exp(y(+1)) / exp(k) + (1-delta)  );

    exp(y) = exp(c) + exp(k) - (1-delta)*exp(k(-1));

    exp(y) = exp(a) * exp(k(-1))^alpha * exp(n)^(1-alpha);

    a = rho_a * a(-1) + e_a;

end;
```

The Model Block

- No need to write down expectations operator.
  - We later declare whether the economy is stochastic or not.
- Good habits:
  - Comment before every equation what it means.
  - Leave plenty of space between equations and break up long equations.

The Initval Block

- Dynare needs to now what values the variable take in steady state.
- Can provide an initial guess for Dynare and let it compute the steady state: This is the initval block.
- The command “resid” will tell you how good this guess is.
- Generally, this only works for small models.
- We will see a more sophisticated way later.

THE INITVAL BLOCK

```
%----------------------------------------------------------
% 4. Steady State
%----------------------------------------------------------

initval;

    y = log(0.907658170398745);
    n = log(0.319336393586796);
    c = log(0.724338047313284);
    k = log(7.332804923418412);
    a = log(1);

end;

resid;
```

Declare the Variances

```
%----------------------------------------------------------
% 5. Shocks
%----------------------------------------------------------

shocks;

    var e_a = sigma_a^2;

end;
```

Computing the Model

```
%-------------------------------------------------------
% 6. Computation
%-------------------------------------------------------
steady;

check;

stoch_simul;
```

Computing the Model

- "steady" which computes the steady state given the initial values.
- "check" determines whether the solution is unique.
- "stoch_simul(options)" computes the dynamics of the model.
- Most important options:
  - order = 1, 2, or 3: this tells Dynare the order of the (log) approximation. The default is a second order approximation. Hence, typing "order=1" will have it do a linear approximation.
  - irf = integer: this will determine the horizon if the IRFs. The default is 40.
  - hpfiler = [smoothing parameter]: this will produce theoretical moments (variances, covariances, autocorrelations) after HP filtering the data (the default is to apply no filter to the data).

Dynare Checks

```
Residuals of the static equations:

Equation number 1 : 0 : 1
Equation number 2 : 0 : 2
Equation number 3 : 0 : 3
Equation number 4 : 0 : 4
Equation number 5 : 0 : 5
Equation number 6 : 0 : a
```

Dynare Checks

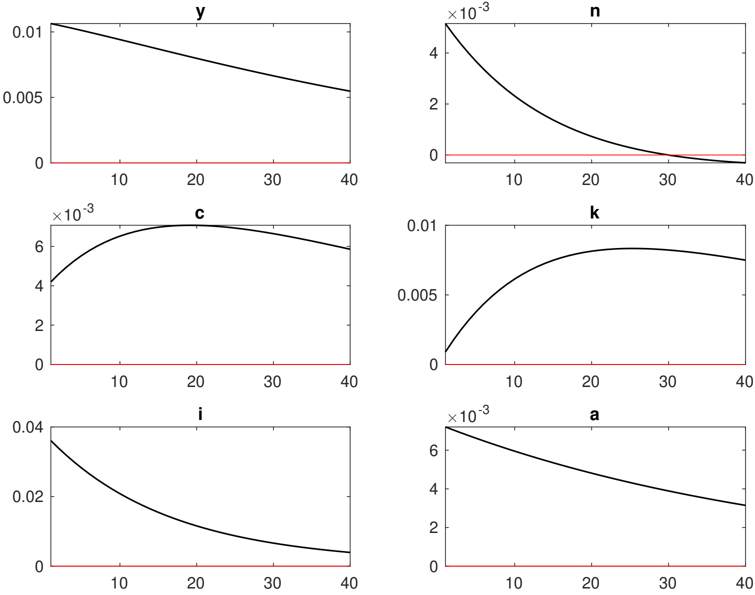
```
EIGENVALUES:
         Modulus             Real        Imaginary

          0.9361           0.9361                0
           0.979            0.979                0
           1.086            1.086                0
       1.304e+16        1.304e+16                0


There are 2 eigenvalue(s) larger than 1 in modulus
for 2 forward-looking variable(s)

The rank condition is verified.
```

Policy Functions

POLICY AND TRANSITION FUNCTIONS

	y	n	c	k	i
Constant	-0.09688	-1.14151	-0.32249	1.99235	-1.696
k(-1)	0.10854	-0.33718	0.53002	0.93607	-1.556
a(-1)	1.44609	0.70063	0.57029	0.12266	4.906
e_a	1.47711	0.71566	0.58252	0.12529	5.011

Impulse Response Functions



- By default Dynare will produce the IRFs.

THEORETICAL MOMENTS

THEORETICAL MOMENTS (HP filter, lambda = 1600)

VARIABLE	MEAN	STD. DEV.	VARIANCE
y	-0.0969	0.0139	0.0002
n	-1.1415	0.0067	0.0000
c	-0.3225	0.0060	0.0000
k	1.9924	0.0041	0.0000
i	-1.6965	0.0469	0.0022
a	0.0000	0.0094	0.0001

MATRIX OF CORRELATIONS (HP filter, lambda = 1600)

Variables	y	n	c	k	i	a
y	1.0000	0.9713	0.9435	0.3899	0.9856	0.9995
n	0.9713	1.0000	0.8375	0.1595	0.9975	0.9784
c	0.9435	0.8375	1.0000	0.6731	0.8739	0.9323
k	0.3899	0.1595	0.6731	1.0000	0.2287	0.3600
i	0.9856	0.9975	0.8739	0.2287	1.0000	0.9906
a	0.9995	0.9784	0.9323	0.3600	0.9906	1.0000

Theoretical Moments

COEFFICIENTS OF AUTOCORRELATION (HP filter, lambda = 1600)

Order	1	2	3	4	5
y	0.7237	0.4877	0.2905	0.1295	0.0017
n	0.7071	0.4615	0.2599	0.0986	-0.0265
c	0.7880	0.5895	0.4090	0.2490	0.1111
k	0.9592	0.8609	0.7249	0.5677	0.4025
i	0.7090	0.4645	0.2634	0.1021	-0.0233
a	0.7199	0.4816	0.2834	0.1223	-0.0048

- Can compare to our business cycle facts lecture.

Where is this Stuff?

- Dynare stores IRFs under the name "x_e" for variable x's response to shock e.
  - So y_e_a is the IRF of output to the TFP shock in our model.
- Lots of other good stuff is stored in M_ and oo_.
  - M_ is a structure with information about the model.
  - oo_ is a structure with information about the simulation.

Looping over Parameters

- We will often want to loop over parameters.
- Write a separate matlab file called RBCloop.m:

```
clear all;
close all;

alpha  = 1/3;
beta  = 0.99;
gamma = 1;
chi = 1;
phi  = 1;
delta = 0.025;
rho_a = 0.95;
sigma_a = 0.01;

save paramfile alpha beta gamma chi phi delta rho_a sigma_a;
```

Looping over Parameters

```
for rho_a = [0.8,0.9,0.95]

save paramfile alpha beta gamma chi phi delta rho_a sigma_a;

dynare RBCloop noclearall nolog;

% save your output here, e.g. impulse response functions

end
```

Looping over Parameters

- Save previous mod file as RBC_looping.mod and change the following code:

```
%----------------------------------------------------------
% 2. Calibration
%----------------------------------------------------------
load paramfile;

set_param_value('alpha',alpha);
set_param_value('beta',beta);
set_param_value('gamma',gamma);
set_param_value('delta',delta);
set_param_value('chi',chi);
set_param_value('phi',phi);
set_param_value('rho_a',rho_a);
set_param_value('sigma_a',sigma_a);
```

Computing the Steady State

- You can supply dynare with a matlab file that computes the steady state.
  - Much faster and more robust, especially for larger models.
  - Especially useful when you loop over parameters.
  - Can also be used to find the parameters that match certain moments.
- For your [filename].mod file it should be called [filename]_steadystate.m. So RBC_steadystate.m in our case.
- It has a predefined structure.

Steady State File

```
[ys,params,check] = RBCloop_steadystate(ys,exo,M_,options_)

% initialize indicator
check = 0;

% same order as parameter declaration
alpha = M_.params(1);
beta = M_.params(2);
gamma = M_.params(3);
chi = M_.params(4);
phi = M_.params(5);
delta = M_.params(6);
rho_a = M_.params(7);
sigma_a = M_.params(8);
```

Steady State File (cont'd)

```
[add equations to compute steady state here]

%% end own model equations
NumberOfParameters = M_.param_nbr;
params=NaN(NumberOfParameters,1);
for iter = 1:length(M_.params) %update parameters set in the f
  eval([ 'params(' num2str(iter) ') = ' M_.param_names{iter} '
end

NumberOfEndogenousVariables = M_.orig_endo_nbr; %auxiliary var
for ii = 1:NumberOfEndogenousVariables
  varname = M_.endo_names{ii};
  eval(['ys(' int2str(ii) ') = ' varname ';']);
end

end
```

Other Tips

- When you build a model start small.
  - Add one element at a time and make sure nothing breaks.
  - Write a test script.
- Use version control (e.g. Git + Github).
- Dynare includes example files in Dynare/[version]/examples.

Next Steps

- Compare model to the data.
- Extend the model.
  - Solve decentralized RBC model: Section.
  - Solve New Keynesian model.
- Estimate the model.