Title

stata.com

Intro 3e — Nonlinear New Classical model

Description Remarks and examples Reference Also see

Description

This introduction describes a nonlinear New Classical model, estimates some of its parameters, and explores the model's impulse–response functions under two different parameter settings.

Remarks and examples

stata.com

Remarks are presented under the following headings:

The model

Equations (1)–(8) specify a variant on the New Classical model studied in King and Rebelo (1999). It includes equations for output Y_t, consumption C_t, investment I_t hours worked H_t, the interest rate R_t, the real wage W_t, the capital stock K_t, and productivity Z_t. Model variables must be weakly stationary. The model contains six parameters: α, β, χ, δ, ρ, and σ.

$$\frac{1}{C_t} = \beta E_t \left\{ \left(\frac{1}{C_{t+1}} \right) (1 + R_{t+1} - \delta) \right\} \tag{1}$$

$$\chi H_t = \frac{W_t}{C_t} \tag{2}$$

$$Y_t = C_t + I_t \tag{3}$$

$$Y_t = Z_t K_t^{\alpha} H_t^{1-\alpha} \tag{4}$$

$$R_t = \alpha \frac{Y_t}{K_t} \tag{5}$$

$$W_t = (1 - \alpha) \frac{Y_t}{H_t} \tag{6}$$

$$K_{t+1} = I_t + (1 - \delta) K_t \tag{7}$$

$$\ln(Z_{t+1}) = \rho \ln(Z_t) + e_{t+1} \tag{8}$$

Equation (1) is a consumption Euler equation that links consumption in the current period to expected future consumption and the expected future interest rate. Equation (2) is a labor supply equation that links hours worked to the wage and consumption. Equation (3) is a standard national income accounting identity, stating that output is split between consumption and investment. Equation (4) is an output supply equation or production function, stating that output is produced by combining capital K_t and labor H_t at productivity level Z_t. Equation (5) is a capital demand curve. Equation (6) is a labor demand curve. Equation (7) is the capital accumulation process. Equation (8) specifies a stochastic process for productivity. The stochastic shock e_{t+1} is i.i.d. normal with mean zero and standard deviation σ_z.

The model includes six parameters. β is the discount factor reflecting a preference for current consumption relative to future consumption. α is a production parameter. χ is a preference parameter. δ is the depreciation rate. ρ measures the persistence of the stochastic productivity process. Finally, σ is the standard deviation of the innovations to the productivity process.

Parameter estimation

In this example, we fix some parameters at prespecified values and estimate others. Once the parameters are estimated, we analyze the model by inspecting its steady state, low-order moments, policy and transition matrices, and impulse–response functions.

We use the growth rate of output as the empirical counterpart to y_t. For the calibrated parameters, we set $\alpha = 0.33$, $\beta = 0.99$, $\delta = 0.025$, and $\chi = 2$, which follows King and Rebelo (1999). We will estimate (ρ, σ_z). We first bring in the data.

```
. use https://www.stata-press.com/data/r18/usmacro2
(Federal Reserve Economic Data - St. Louis Fed, 2017-01-15)
```

We next set up all the constraints

```
. constraint 1 _b[alpha] = 0.33
. constraint 2 _b[beta]  = 0.99
. constraint 3 _b[delta] = 0.025
. constraint 4 _b[chi]   = 2
```

and estimate the remaining parameters,

```
. dsgenl (1/c = {beta}*(1/F.c)*(1+F.r-{delta}))
>        ({chi}*h = w/c)
>        (y   = c + i)
>        (y   = z*k^{alpha}*h^(1-{alpha}))
>        (r   = {alpha}*y/k)
>        (w   = (1-{alpha})*y/h)
>        (F.k = i + (1-{delta})*k)
>        (ln(F.z) = {rho}*ln(z))
> , observed(y) unobserved(c i r w h) exostate(z) endostate(k) constraint(1/4)
Solving at initial parameter vector ...
Checking identification ...
(setting technique to bfgs)
Iteration 0:  Log likelihood = -955.44919
Iteration 1:  Log likelihood = -857.31841  (backed up)
Iteration 2:  Log likelihood = -850.75913  (backed up)
Iteration 3:  Log likelihood = -850.75913  (backed up)
Iteration 4:  Log likelihood = -827.82709
Iteration 5:  Log likelihood = -827.82709  (backed up)
Iteration 6:  Log likelihood = -827.82709  (backed up)
BFGS stepping has contracted, resetting BFGS Hessian
Iteration 7:  Log likelihood = -812.65536
Iteration 8:  Log likelihood = -809.74135  (backed up)
Iteration 9:  Log likelihood = -764.90349
(switching technique to nr)
Iteration 10: Log likelihood = -764.90349  (not concave)
Iteration 11: Log likelihood =  -702.6619
Iteration 12: Log likelihood = -691.50282
Iteration 13: Log likelihood = -689.88687  (not concave)
Iteration 14: Log likelihood = -682.85662
Iteration 15: Log likelihood = -653.96651
Iteration 16: Log likelihood = -639.87678
Iteration 17: Log likelihood = -639.39734
Iteration 18: Log likelihood = -639.39684
Iteration 19: Log likelihood = -639.39684
```

```
First-order DSGE model
Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -639.39684
 ( 1)  [/structural]alpha = .33
 ( 2)  [/structural]beta = .99
 ( 3)  [/structural]delta = .025
 ( 4)  [/structural]chi = 2
```

y	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.99	(constrained)				
delta	.025	(constrained)				
chi	2	(constrained)				
alpha	.33	(constrained)				
rho	.3133483	.0614696	5.10	0.000	.1928701	.4338265
sd(e.z)	2.287682	.1036018			2.084627	2.490738

We specified four options related to the model structure. The `exostate(z)` option lists the state variable that is subject to shocks, `z`. The `endostate(k)` option specifies that `k` is a state variable without shocks. The `observed(y)` option specifies `y` as an observed control variable. The `unobserved(c i r w h)` option specifies those variables as unobserved, or latent, control variables. The remaining option we specified was `constraint(1/4)`, which applies the constraints we set up previously. We find that the estimated productivity process is mildly persistent, $\rho = 0.31$. The standard deviation of the productivity shock is about 2.3, or about 2%.

Steady state

In the absence of shocks, the variables in a nonlinear DSGE model converge to a steady-state position in which variables are constant through time. `dsgenl` finds the steady state of the model at the estimated parameter values. You can view the steady state with `estat steady`.

```
. estat steady
Location of model steady-state
```

	Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
k	18.75991	.	.	.	.	.
z	1	.	.	.	.	.
c	1.526432	.	.	.	.	.
i	.4689977	.	.	.	.	.
r	.035101	.	.	.	.	.
w	2.02027	.	.	.	.	.
h	.6617621	.	.	.	.	.
y	1.99543	.	.	.	.	.

```
Note: Standard errors reported as missing for constrained steady-state values.
```

All the standard errors for the estimated location of the steady state are missing. This is because of the constraints we placed on some of the model parameters. We estimated the persistence of the productivity state variable and the standard deviation of the shock to productivity. These dynamic parameters have no effect on the steady state. Instead, the steady state is a function of the model's static parameters. Because we constrained all the static parameters, the steady-state location is not subject to uncertainty, and the standard errors are all missing. Put another way, the location of the

steady state is determined entirely by parameters that we have constrained in the model and by the model's structure itself.

Model-implied covariances

The model's state-space matrices generate predictions for the variances and covariances of the model variables.

```
. estat covariance
Estimated covariances of model variables
```

	Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
c						
var(c)	.870567	.1701705	5.12	0.000	.537039	1.204095
cov(c,i)	3.39887	.5499555	6.18	0.000	2.320977	4.476763
cov(c,r)	.0352466	.0232027	1.52	0.129	-.0102298	.080723
cov(c,w)	1.167688	.2145175	5.44	0.000	.7472416	1.588135
cov(c,h)	.2971211	.0448135	6.63	0.000	.2092883	.3849539
cov(c,y)	1.464809	.2590251	5.66	0.000	.9571294	1.972489
i						
var(i)	196.9607	19.01966	10.36	0.000	159.6829	234.2386
cov(i,r)	48.87007	4.692839	10.41	0.000	39.67227	58.06786
cov(i,w)	26.14587	2.644813	9.89	0.000	20.96213	31.32961
cov(i,h)	22.747	2.18294	10.42	0.000	18.46852	27.02549
cov(i,y)	48.89287	4.818514	10.15	0.000	39.44876	58.33699
r						
var(r)	12.92995	1.273471	10.15	0.000	10.434	15.42591
cov(r,w)	5.774216	.5477555	10.54	0.000	4.700635	6.847797
cov(r,h)	5.73897	.5520461	10.40	0.000	4.656979	6.82096
cov(r,y)	11.51319	1.099565	10.47	0.000	9.358078	13.66829
w						
var(w)	4.103074	.4796048	8.56	0.000	3.163066	5.043082
cov(w,h)	2.935386	.2902517	10.11	0.000	2.366503	3.504269
cov(w,y)	7.03846	.7632272	9.22	0.000	5.542562	8.534358
h						
var(h)	2.638265	.2521725	10.46	0.000	2.144016	3.132514
cov(h,y)	5.573651	.5419094	10.29	0.000	4.511528	6.635774
y						
var(y)	12.61211	1.298179	9.72	0.000	10.06773	15.15649

One object of interest is the relative volatility of a model variable, defined as the standard deviation of that variable compared with the standard deviation of a reference variable. We can calculate the volatility of investment (model variable `i`) relative to output (model variable `y`) with `nlcom`.

Before we can use `nlcom`, we need to add the `post` option to our `estat covariance` command. With this option, the results are stored in `e()`, where `nlcom` can access them. First, however, we save the `dsgenl` estimates using `estimates store`.

```
. estimates store dsgenl
. quietly estat covariance, post
```

We can now refer to the variances of investment and output as `_b[i:var(i)]` and `_b[y:var(y)]`. We use `nlcom` to compute the volatility of investment relative to output.

```
. nlcom sqrt(_b[i:var(i)] / _b[y:var(y)])
       _nl_1: sqrt(_b[i:var(i)] / _b[y:var(y)])
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
_nl_1	3.951809	.0303923	130.03	0.000	3.892241	4.011377

The model predicts that investment is about four times as volatile as output.

Policy and transition matrices

The model's state-space form expresses the control variables as functions of the state variables alone. It solves the system of equations locally near the steady state, resolving the expectations of future variables via the rational expectations assumption. The entries in the state-space matrices are interpretable as the contemporaneous effect of a one-unit change in the state variable on the control variable.

```
. estimates restore dsgenl
(results dsgenl are active now)
. estat policy
Policy matrix
```

		Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
c							
	k	.5530748	.	.	.	.	.
	z	.1006471	.0040593	24.79	0.000	.092691	.1086031
i							
	k	-.8741569	.	.	.	.	.
	z	5.854704	.0219119	267.19	0.000	5.811758	5.897651
r							
	k	-.782376	.	.	.	.	.
	z	1.453057	.0020449	710.58	0.000	1.44905	1.457065
w							
	k	.3853494	.	.	.	.	.
	z	.7768523	.0010072	771.31	0.000	.7748782	.7788263
h							
	k	-.1677254	.	.	.	.	.
	z	.6762052	.0030521	221.56	0.000	.6702232	.6821872
y							
	k	.217624	.	.	.	.	.
	z	1.453057	.0020449	710.58	0.000	1.44905	1.457065

```
Note: Standard errors reported as missing for constrained policy matrix
      values.
```

The standard errors presented here again require some interpretation. The standard errors for the impact effect of `k` on all model variables are missing. These impact effects depend entirely on the parameters we constrained, so their values are fixed entirely by the structure of the model.

The state variables are correlated in this model. The capital equation shown in (7) specifies that future capital is a function of the current capital stock and current investment. In turn, investment depends on the current capital stock and on productivity; hence, the future capital stock depends on both the current capital stock and current productivity. We can see these relationships with `estat transition`.

```
. estat transition
Transition matrix of state variables
```

		Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
F.k							
	k	.9531461	.	.	.	.	.
	z	.1463676	.0005478	267.19	0.000	.1452939	.1474413
F.z							
	k	0	(omitted)				
	z	.3133483	.0614696	5.10	0.000	.1928701	.4338265

```
Note: Standard errors reported as missing for constrained transition matrix
      values.
```

The "own" effects (`k` on `F.k` and `z` on `F.z`) are persistence parameters. The other reported effects show how a change in one state variable contemporaneously affects another. A change in the capital stock does not affect future productivity, but a change in productivity does affect the future capital stock. A 1% increase in productivity raises the future capital stock by 0.15%.

Impulse responses

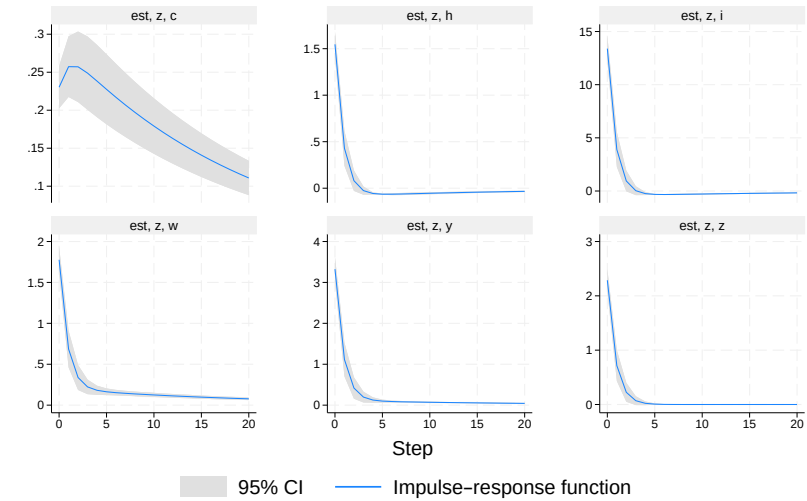
The policy and transition matrices display impact effects. Because of the dynamic structure of the state, a shock to a state variable has lasting effects on both the state and the controls. An impulse–response function traces out the full dynamic effect of a shock. When state variables are correlated, a shock to one state variable will (either immediately or eventually) lead to movements in another state variable.

We use the `irf` commands to trace out the effect of a shock to productivity on itself and on each of the control variables in our model.

```
. irf set rbcirf, replace
(file rbcirf.irf created)
(file rbcirf.irf now active)

. irf create est, step(20) replace
(irfname est not found in rbcirf.irf)
(file rbcirf.irf updated)

. irf graph irf, impulse(z) response(y c i h w z) byopts(yrescale)
```



Graphs by irfname, impulse variable, and response variable

All model variables rise on a shock to productivity. For most model variables, the increase is short lived, and they return to their long-run values within five periods. Consumption is the exception (upper left panel). It rises smoothly for several periods, then gradually falls back to its long-run value.

Sensitivity analysis

In this section, we will explore the effect of changing some model parameters on the impulse responses. We first type the `dsgenl` command without arguments to replay the current estimates.

```
. dsgenl
First-order DSGE model
Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -639.39684
 ( 1)  [/structural]alpha = .33
 ( 2)  [/structural]beta = .99
 ( 3)  [/structural]delta = .025
 ( 4)  [/structural]chi = 2
```

y	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.99	(constrained)				
delta	.025	(constrained)				
chi	2	(constrained)				
alpha	.33	(constrained)				
rho	.3133483	.0614696	5.10	0.000	.1928701	.4338265
sd(e.z)	2.287682	.1036018			2.084627	2.490738

We can solve the model using different parameter values and analyze how the model behaves across the two scenarios. It is common to examine a model's behavior for multiple parameter settings, for example, for different settings for the persistence of a shock or for different settings of preference, technology, or policy parameters. The simplest way to solve using a new set of parameters is to use the `from()` and `solve` options. First, store the current parameter vector in a Stata matrix.

```
. matrix b = e(b)
```

Next, change the entries in the matrix to the new desired parameters. In this case, we want to look at the response of the model to a shock when productivity is more persistent than was estimated, so we set $\rho = 0.6$.

```
. matrix b[1,5] = 0.6
```

Next, we solve the model again. In the following command, note the use of the `from(b)` and `solve` options.

```
. dsgenl (1/c = {beta}*(1/F.c)*(1+F.r-{delta}))
>        ({chi}/(1-h) = w/c)
>        (y   = c + i)
>        (y   = z*k^{alpha}*h^(1-{alpha}))
>        (w   = (1-{alpha})*y/h)
>        (r   = {alpha}*y/k)
>        (F.k = i + (1-{delta})*k)
>        (ln(F.z) = {rho}*ln(z))
>        , observed(y) unobserved(c i r w h) exostate(z) endostate(k)
>        from(b) solve noidencheck
Solving at initial parameter vector ...

First-order DSGE model

Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -653.63973
```

y	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.99	.	.	.	.	.
delta	.025	.	.	.	.	.
chi	2	.	.	.	.	.
alpha	.33	.	.	.	.	.
rho	.6	.	.	.	.	.
sd(e.z)	2.287682	.			.	.

```
Note: Skipped identification check.
Note: Model solved at specified parameters; maximization options ignored.
```

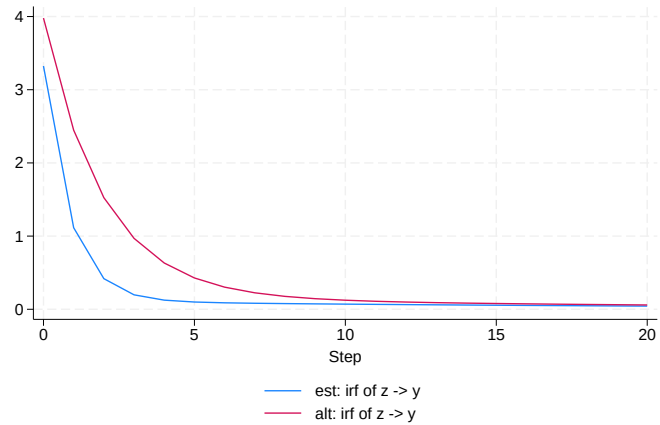
We passed the parameter vector in using `from()`, and we used the `solve` option to prevent estimation. Because we specified the `solve` option, the estimation table simply repeats the values we fed in with `from()`. Of course, because we did not estimate, we do not get standard errors. When we use `solve`, the point of the analysis is always in the `estat` and `irf` commands that come after.

We create a new set of impulse responses.

```
. irf create alt, step(20) replace
(irfname alt not found in rbcirf.irf)
(file rbcirf.irf updated)
```

We graph the response of `y` to a `z` impulse across the two models.

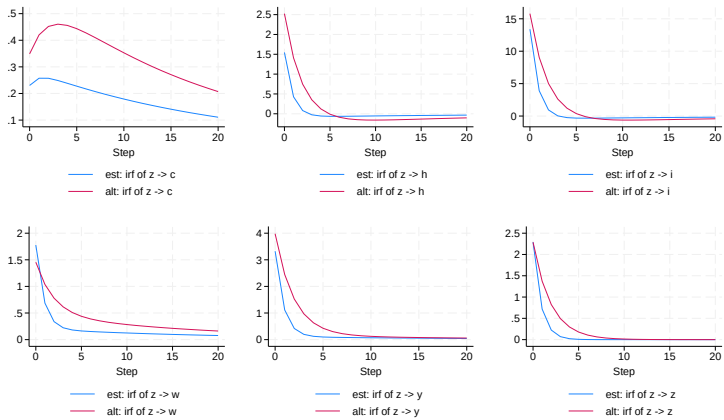
```
. irf ograph (est z y irf) (alt z y irf)
```



We see that output increases more and takes longer to return to its long-run value when we set ρ to 0.60 instead of its estimated value of 0.31.

We are not interested only in the response of `y` to a shock in `z`. We can also plot the response of `c`, `h`, `i`, `w`, `y`, and `z` to an impulse to `z` in panels of a combined graph. We will accomplish this task with a `foreach` loop.

```
. foreach v in c h i w y z {
  2.         irf ograph (est z ‘v’ irf) (alt z ‘v’ irf), name(‘v’) nodraw
  3. }
. graph combine c h i w y z
```



The model with $\rho = 0.60$ shows more persistence overall than the estimated value of $\rho = 0.31$, as expected. For variables such as consumption and wages, the amplitude of the response is amplified as well.

Reference

King, R. G., and S. T. Rebelo. 1999. Resuscitating real business cycles. In *Handbook of Macroeconomics: Volume 1A*, ed. J. B. Taylor and M. Woodford, 927–1007. New York: Elsevier. https://doi.org/10.1016/S1574-0048(99)10022-3.

Also see

[DSGE] **Intro 1** — Introduction to DSGEs

[DSGE] **Intro 3b** — New Classical model

[DSGE] **Intro 3d** — Nonlinear New Keynesian model

[DSGE] **Intro 3f** — Stochastic growth model

[DSGE] **dsgenl** — Nonlinear dynamic stochastic general equilibrium models

[DSGE] **dsgenl postestimation** — Postestimation tools for dsgenl