STATA DYNAMIC STOCHASTIC GENERAL EQUILIBRIUM MODELS REFERENCE MANUAL

RELEASE 18

A Stata Press Publication
StataCorp LLC
College Station, Texas

Version 18

Published by Stata Press, 4905 Lakeway Drive, College Station, Texas 77845

ISBN-10: 1-59718-377-6
ISBN-13: 978-1-59718-377-2

The suggested citation for this software is

> StataCorp. 2023. *Stata 18*. Statistical software. StataCorp LLC.

The suggested citation for this manual is

> StataCorp. 2023. *Stata 18 Dynamic Stochastic General Equilibrium Models Reference Manual*. College Station, TX: Stata Press.

Contents

Cross-referencing the documentation

When reading this manual, you will find references to other Stata manuals, for example, **[U] 27 Overview of Stata estimation commands**; [R] **regress**; and [D] **reshape**. The first example is a reference to chapter 27, *Overview of Stata estimation commands*, in the *User's Guide*; the second is a reference to the `regress` entry in the *Base Reference Manual*; and the third is a reference to the `reshape` entry in the *Data Management Reference Manual*.

All the manuals in the Stata Documentation have a shorthand notation:

[GSM]	*Getting Started with Stata for Mac*
[GSU]	*Getting Started with Stata for Unix*
[GSW]	*Getting Started with Stata for Windows*
[U]	*Stata User's Guide*
[R]	*Stata Base Reference Manual*
[ADAPT]	*Stata Adaptive Designs: Group Sequential Trials Reference Manual*
[BAYES]	*Stata Bayesian Analysis Reference Manual*
[BMA]	*Stata Bayesian Model Averaging Reference Manual*
[CAUSAL]	*Stata Causal Inference and Treatment-Effects Estimation Reference Manual*
[CM]	*Stata Choice Models Reference Manual*
[D]	*Stata Data Management Reference Manual*
[DSGE]	*Stata Dynamic Stochastic General Equilibrium Models Reference Manual*
[ERM]	*Stata Extended Regression Models Reference Manual*
[FMM]	*Stata Finite Mixture Models Reference Manual*
[FN]	*Stata Functions Reference Manual*
[G]	*Stata Graphics Reference Manual*
[IRT]	*Stata Item Response Theory Reference Manual*
[LASSO]	*Stata Lasso Reference Manual*
[XT]	*Stata Longitudinal-Data/Panel-Data Reference Manual*
[META]	*Stata Meta-Analysis Reference Manual*
[ME]	*Stata Multilevel Mixed-Effects Reference Manual*
[MI]	*Stata Multiple-Imputation Reference Manual*
[MV]	*Stata Multivariate Statistics Reference Manual*
[PSS]	*Stata Power, Precision, and Sample-Size Reference Manual*
[P]	*Stata Programming Reference Manual*
[RPT]	*Stata Reporting Reference Manual*
[SP]	*Stata Spatial Autoregressive Models Reference Manual*
[SEM]	*Stata Structural Equation Modeling Reference Manual*
[SVY]	*Stata Survey Data Reference Manual*
[ST]	*Stata Survival Analysis Reference Manual*
[TABLES]	*Stata Customizable Tables and Collected Results Reference Manual*
[TS]	*Stata Time-Series Reference Manual*
[I]	*Stata Index*
[M]	*Mata Reference Manual*

Title

Intro — Introduction to DSGE manual

Description Remarks and examples Also see

Description

DSGE stands for dynamic stochastic general equilibrium. DSGE models are multivariate time-series models that are used in economics, in particular, macroeconomics, for policy analysis and forecasting. These models are systems of equations that are typically derived from economic theory. As such, the parameters are often directly interpretable based on theory. DSGE models are unique in that equations in the system allow current values of variables to depend not only on past values but also on expectations of future values.

The `dsgenl` command estimates parameters of nonlinear DSGE models.

The `dsge` command estimates parameters of linear DSGE models.

Remarks and examples

We recommend that you read this manual beginning with [DSGE] **Intro 1** and then continue with the remaining introductions. In these introductions, we will introduce DSGE models, show you how to use the `dsgenl` and `dsge` commands, walk you through worked examples of classic models, and present solutions to common stumbling blocks.

[DSGE] **Intro 1** and [DSGE] **Intro 2** are essential reading. Read them first. Here you will find an overview of DSGE models, descriptions of concepts used throughout the manual, discussion of assumptions, a first worked example, and an introduction to the syntax.

[DSGE] **Intro 1**	Introduction to DSGEs
[DSGE] **Intro 2**	Learning the syntax

[DSGE] **Intro 3** focuses on classical DSGE models. It includes a series of examples that illustrate model solution, model estimation, and postestimation procedures for simple variants of common models.

[DSGE] **Intro 3**	Classic DSGE examples
[DSGE] **Intro 3a**	New Keynesian model
[DSGE] **Intro 3b**	New Classical model
[DSGE] **Intro 3c**	Financial frictions model
[DSGE] **Intro 3d**	Nonlinear New Keynesian model
[DSGE] **Intro 3e**	Nonlinear New Classical model
[DSGE] **Intro 3f**	Stochastic growth model

[DSGE] **Intro 4** discusses some features commonly found in DSGE models and how to specify models with those features to `dsge` and `dsgenl`. The structural equations of the DSGE model must have a specific structure so that the model can be solved. Often, DSGE models are written using intuitive forms that do not have this structure. These intuitive forms can be rewritten in a logically equivalent form that has the structure required for solution. [DSGE] **Intro 4** provides an overview of this topic and examples demonstrating solutions.

[DSGE] **Intro 4**	Writing a DSGE in a solvable form
[DSGE] **Intro 4a**	Specifying a shock on a control variable
[DSGE] **Intro 4b**	Including a lag of a control variable
[DSGE] **Intro 4c**	Including a lag of a state variable
[DSGE] **Intro 4d**	Including an expectation dated by more than one period ahead
[DSGE] **Intro 4e**	Including a second-order lag of a control
[DSGE] **Intro 4f**	Including an observed exogenous variable
[DSGE] **Intro 4g**	Correlated state variables

[DSGE] **Intro 5**–[DSGE] **Intro 8** discuss technical issues. These introductions are essential reading, even though they are last.

[DSGE] **Intro 5**	Stability conditions
[DSGE] **Intro 6**	Identification
[DSGE] **Intro 7**	Convergence problems
[DSGE] **Intro 8**	Wald tests vary with nonlinear transforms

[DSGE] **Intro 9**, [DSGE] **Intro 9a**, and [DSGE] **Intro 9b** discuss Bayesian estimation.

[DSGE] **Intro 9**	Bayesian estimation
[DSGE] **Intro 9a**	Bayesian estimation of a New Keynesian model
[DSGE] **Intro 9b**	Bayesian estimation of stochastic growth model

The main command entries are references for syntax and implementation details. All the examples are in the introductions discussed above.

[DSGE] **dsge**	Linear dynamic stochastic general equilibrium models
[DSGE] **dsge postestimation**	Postestimation tools for dsge
[DSGE] **dsgenl**	Nonlinear dynamic stochastic general equilibrium models
[DSGE] **dsgenl postestimation**	Postestimation tools for dsgenl
[DSGE] **estat covariance**	Display estimated covariances of model variables
[DSGE] **estat policy**	Display policy matrix
[DSGE] **estat stable**	Check stability of system
[DSGE] **estat steady**	Display steady state of nonlinear DSGE model
[DSGE] **estat transition**	Display state transition matrix

[BAYES] **bayes: dsge**	Bayesian linear dynamic stochastic general equilibrium models
[BAYES] **bayes: dsgenl**	Bayesian nonlinear dynamic stochastic general equilibrium models
[BAYES] **bayes: dsge postestimation**	Postestimation tools for bayes: dsge and bayes: dsgenl
[BAYES] **bayesirf**	Bayesian IRFs, dynamic-multiplier functions, and FEVDs

Also see

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

Title

Intro 1 — Introduction to DSGEs

Description Remarks and examples References Also see

Description

In this entry, we introduce DSGE models and the two commands that estimate the parameters of DSGE models, `dsgenl` and `dsge`. For Bayesian estimation, see [BAYES] **bayes: dsge** and [BAYES] **bayes: dsgenl**. We begin with an overview of DSGE models. We then illustrate the process of DSGE modeling by working an example. In this example, we describe a model in its original nonlinear form and estimate its parameters using `dsgenl`. We also write the model in a corresponding linearized form and estimate the parameters using `dsge`. We conclude by showing how this example fits into the general DSGE modeling framework.

Remarks and examples

Remarks are presented under the following headings:

Introduction to DSGE models

DSGE models are models for multiple time series used in macroeconomics and finance. These models are systems of equations that are motivated by economic theory and in which expectations of future values of variables play an important role. Because these models come from theory, the parameters of these models can typically be directly interpreted in terms of the motivating theory. DSGE models are used for macroeconomic policy analysis and forecasting.

In DSGE models, individuals' actions are summarized by decision rules that take the form of nonlinear systems of dynamic equations. These decision rules often come from dynamic stochastic optimization problems. A DSGE model consists of these decision rules, plus any aggregation conditions, resource or budget constraints, and stochastic processes for exogenous variables.

Because the model's equations are the solution to dynamic optimization problems, model equations can feature expectations of future variables. These expectations are endogenous. In DSGE models, expectations of future values of variables correspond to their conditional mean as implied by the model. In other words, individuals' expectations of future values are correct, on average. Such expectations are said to be model-consistent expectations or rational expectations.

There are three kinds of variables in DSGE models: control variables, state variables, and shocks. The terminology is taken from the state-space and optimal control literatures. In DSGE models, the concepts of exogeneity and endogeneity are understood relative to a time period. A state variable is fixed, or exogenous, in a given time period. The system of equations then determines the value of the state variable one period in the future. On the other hand, the system of equations determines the value of a control variable in the current time period. Control variables in a DSGE model can be either observed or unobserved. State variables are always unobserved.

DSGE models can be written in multiple forms. The model may consist of equations that are nonlinear both in the variables and in the parameters. Such a model is said to be nonlinear. A DSGE model is said to be linear, or linearized, when the model equations are linear in the variables. Linear models may still be nonlinear in their parameters.

Nonlinear DSGE models are commonly linearized prior to analysis. In Stata, you have two choices for handling the linearization of a nonlinear model. You can write your model in its nonlinear form and use the `dsgenl` command. `dsgenl` will take a linear approximation to the nonlinear model for you. Alternatively, you can derive the linear form of the model and use the `dsge` command. Although linearizing the equations yourself requires extra effort, there can be advantages. Linearized DSGE models are easier to manipulate than nonlinear ones. For instance, it is easier to include additional lag terms in a linearized model than in a nonlinear one.

After linearization, the DSGE model must be solved prior to estimation. In any analysis of simultaneous equations systems, to solve a model means to write the model's endogenous variables as functions of its exogenous variables. In DSGE models, the analogous concept is to write the model's control variables in terms of its state variables. The model's solution consists of a system of equations relating the control variables to the state variables and a system of equations describing the evolution of the state variables over time. The solution to a DSGE model thus takes the form of a state-space model. The solution to a DSGE model is a crucial object for both estimation and analyses after estimation. Both the likelihood function and the impulse–response functions are formed from the model solution.

`dsgenl` solves and estimates the parameters of nonlinear DSGE models. It linearizes the model by taking a first-order approximation to the model's equations at the model's steady state, solves the linearized model, and estimates the parameters of the model using the linearized solution.

`dsge` solves and estimates the parameters of linearized DSGE models.

For Bayesian estimation, see [DSGE] **Intro 9**. `bayes: dsge` fits Bayesian linear DSGE models, and `bayes: dsgenl` fits Bayesian nonlinear DSGE models.

General introductions to DSGE modeling are available in Ljungqvist and Sargent (2018) and Woodford (2003). Canova (2007) and DeJong and Dave (2011) describe parameter estimation using DSGE models. Fernández-Villaverde, Rubio-Ramírez, and Schorfheide (2016) provide an overview of solution and estimation strategies for nonlinear DSGE models.

An example: A nonlinear DSGE model

Writing down nonlinear DSGEs

Consider the following nonlinear model, similar to that in Woodford (2003, chap. 4). The model consists of equations that describe the behavior of households, firms, and a central bank. Interactions among these actors produce a model of inflation, output growth, and the interest rate. Models of this type are popular in academic and policy settings and are used to describe monetary policy.

Household optimization generates an equation that relates current output Y_t to the expected value of a function of tomorrow's output Y_{t+1}, tomorrow's inflation Π_{t+1}, and the current nominal interest rate R_t,

$$\frac{1}{Y_t} = \beta E_t \left(\frac{1}{Y_{t+1}} \frac{R_t}{\Pi_{t+1}} \right) \tag{1}$$

where β is a parameter that captures households' willingness to delay consumption.

Optimization by firms generates an equation that links the current deviation of inflation from its steady state, $\Pi_t - \Pi$, to the expected value of the deviation of inflation from its steady state in the future, $E_t(\Pi_{t+1} - \Pi)$, and to the ratio of actual output, Y_t, to the natural level of output, Z_t,

$$(\Pi_t - \Pi) + \frac{1}{\phi} = \phi \left(\frac{Y_t}{Z_t} \right) + \beta E_t (\Pi_{t+1} - \Pi) \tag{2}$$

where ϕ is a parameter linked to the pricing decision of firms. Firms are not affected by inflation per se; they are affected only by deviations of inflation from its steady-state value.

Finally, there is an equation that describes central bank policy. The central bank adjusts the interest rate in response to inflation and, perhaps, to other factors that we leave unmodeled. The equation for the central bank policy is

$$\frac{R_t}{R} = \left(\frac{\Pi_t}{\Pi} \right)^{1/\beta} U_t \tag{3}$$

where R is the steady-state value of the interest rate and U_t is a state variable that captures all movements in the interest rate that are not driven by inflation.

We make a few modifications to the model equations before estimating the parameters. Woodford (2003) rewrites the model in (1)–(3) by defining $X_t = Y_t/Z_t$ as the output gap. We do the same. Substituting in X_t gives us the three-equation system

$$1 = \beta E_t \left(\frac{X_t}{X_{t+1}} \frac{1}{G_t} \frac{R_t}{\Pi_{t+1}} \right) \tag{4a}$$

$$(\Pi_t - \Pi) + \frac{1}{\phi} = \phi X_t + \beta E_t (\Pi_{t+1} - \Pi) \tag{5a}$$

$$\frac{R_t}{R} = \left(\frac{\Pi_t}{\Pi} \right)^{1/\beta} U_t \tag{6a}$$

where $G_t = Z_{t+1}/Z_t$ is a state variable that captures changes in Z_t.

The model is completed by adding equations describing the evolution of the state variables G_t and U_t. We write them as autoregressive processes in logarithms,

$$\ln(G_{t+1}) = \rho_g \ln(G_t) + \xi_{t+1} \tag{7a}$$

$$\ln(U_{t+1}) = \rho_u \ln(U_t) + e_{t+1} \tag{8a}$$

The variables ξ_{t+1} and ϵ_{t+1} are shocks to the state variables. We now have a complete nonlinear DSGE model. To estimate its parameters, we need to get some data.

Data preparation

There are three requirements of the data used with `dsgenl` and `dsge`. First, the data must be `tsset` so that Stata understands the time structure of your data. Second, the series in your model must have zero mean. This requirement is actually handled for you. Before solving the models, these commands first transform variables into differences from their steady state or, with `dsgenl`, percent deviations from their steady state. The third requirement is that variables in your data must be weakly stationary.

We have data on the price level and the nominal interest rate in `rates.dta`. These data were obtained from the Federal Reserve Economic Database (FRED), which contains many macroeconomic and financial time series; see [D] **import fred**. We can type `tsset` to see that `dateq` has already been set as the quarterly time variable in this dataset.

```
. use https://www.stata-press.com/data/r18/rates2
(Federal Reserve Economic Data - St. Louis Fed, 2017-02-10)
. tsset
Time variable: dateq, 1947q1 to 2017q1
        Delta: 1 quarter
. describe
Contains data from https://www.stata-press.com/data/r18/rates2.dta
 Observations:           281                  Federal Reserve Economic Data -
                                                St. Louis Fed, 2017-02-10
    Variables:             5                  26 Apr 2022 21:22
-------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
-------------------------------------------------------------------------------
datestr         str10   %-10s                 Observation date
daten           int     %td                   Numeric (daily) date
gdpdef          float   %9.0g                 GDP deflator GDPDEF
r               float   %9.0g                 Federal funds rate FEDFUNDS
dateq           int     %tq                   Quarterly date
-------------------------------------------------------------------------------
Sorted by: dateq
```

The dataset includes the price-level variable `gdpdef`. But the model is written in terms of the inflation rate. For quarterly data, the inflation rate is conventionally obtained as 400 times the difference in log of the price level. Therefore, we begin by generating an inflation rate variable `p` by using the `L.` lag operator.

```
. generate p = 400*(ln(gdpdef) - ln(L.gdpdef))
(2 missing values generated)
. label variable p "Inflation rate"
```

We now have inflation and interest rates.

Specifying the model to dsgenl

Equations (4a)–(8a) are the nonlinear DSGE model. The command to estimate the parameters of the system (4a)–(8a) is

```
. dsgenl (1 = {beta}*(x/F.x)*(1/g)*(r/F.p) )                          ///
         (1/{phi} + (p-1) = {phi}*x + {beta}*(F.p-1))                 ///
         ({beta}*r = p^(1/{beta})*u)                                  ///
         (ln(F.u) = {rhou}*ln(u))                                     ///
         (ln(F.g) = {rhog}*ln(g)),                                    ///
         exostate(u g) observed(p r) unobserved(x)
```

Each equation is bound in parentheses. The equations in the command look similar to the equations we wrote down. The equations can be written in nearly any form that is logically equivalent to the model equations. In addition to providing the equations, you must specify which variable plays which role in the model through the options. The option `exostate()` lists all exogenous state variables, those that are subject to shocks. The option `observed()` lists observed control variables. The option `unobserved()` lists unobserved (latent) control variables. A fourth option, `endostate()`, is available for endogenous state variables, those that are not subject to shocks. Each variable must be listed in one and only one of the options. The number of exogenous state variables must be the same as the number of observed control variables.

Parameters are bound in curly braces; this is how Stata distinguishes between parameters and variables. Parameters can appear nonlinearly, they can appear more than once, and they can appear in multiple equations. A parameter repeated across equations is considered the same parameter; for example, in the above model, `beta` appears in each of the first three equations, and it is assumed that there is only one parameter `beta` to be estimated. These cross-equation restrictions are common in DSGE models.

The operator `F.` can be applied to both state and control variables to denote the expected future value of a variable. For example, we typed `F.x` to include x_{t+1} in the model. However, unlike `F.` used in other time-series applications, the value of x_{t+1} is not determined from the value of `x` at time $t+1$ in your data. Instead, it is an expectation of the value of `x` at time $t+1$ based on your model. In fact, the entire model can be written in terms of expectations.

Notice that in (4a) and (5a), the E_t operator appears. This is the statistical expectation operator. It refers to expected value of the term in brackets conditional on information known at time t. In `dsgenl`, this expectation operator is assumed in front of each equation. Hence, the `dsgenl` equation

```
(1 = {beta}*(x/F.x)*(1/g)*(r/F.p))
```

is interpreted as

$$E_t\left(1-\beta\frac{X_t}{X_{t+1}}\frac{1}{G_t}\frac{R_t}{\Pi_{t+1}}\right)=0$$

which is equivalent to what we wrote in (4a).

Parameter estimation and interpretation of nonlinear DSGEs

The model's parameters are estimated by taking a linear approximation to the model's equations, solving the linearized system for its state-space form, and estimating the parameters of the state-space model by maximum likelihood.

```
. dsgenl (1 = {beta}*(x/F.x)*(1/g)*(r/F.p))
>        (1/{phi} + (p-1) = {phi}*x + {beta}*(F.p-1))
>        ({beta}*r = p^(1/{beta})*u)
>        (ln(F.u) = {rhou}*ln(u))
>        (ln(F.g) = {rhog}*ln(g)),
>        exostate(u g) observed(p r) unobserved(x) nolog
Solving at initial parameter vector ...
Checking identification ...

First-order DSGE model

Sample: 1954q3 thru 2016q4                              Number of obs = 250
Log likelihood = -768.09383
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.5112894	.0758003	6.75	0.000	.3627236	.6598553
phi	5.896273	1.653019	3.57	0.000	2.656415	9.136131
rhou	.6989551	.0449119	15.56	0.000	.6109294	.7869809
rhog	.9556397	.0181344	52.70	0.000	.9200968	.9911825
sd(e.u)	2.317581	.29883			1.731885	2.903277
sd(e.g)	.6147482	.0973365			.4239722	.8055242

We see header information describing the sample date range, the number of observations, and the log likelihood.

In the estimation table, we see estimates of model parameters. Some of these parameters have a structural interpretation. Parameter `beta` plays two roles in the model: it is the discount factor in the household and firm equations, and its inverse captures the degree to which the central bank responds to inflation in the interest rate equation. Its point estimate is about one half, so a point estimate for its inverse is about 2.

We use `nlcom` to obtain an estimate of the parameter $1/\beta$ directly.

```
. nlcom 1/_b[beta]
       _nl_1: 1/_b[beta]
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
_nl_1	1.955839	.2899594	6.75	0.000	1.387529	2.524149

Values for this parameter in the literature are typically around 1.5. Our estimated value is about 2, but the common value of 1.5 lies within the confidence interval.

The `phi` parameter is a measure of price adjustment. The remaining four parameters describe the evolution of the model's exogenous state variables. Both state variables are highly persistent, with persistence parameters of 0.70 and 0.96 on `u` and `g`, respectively. The estimated standard deviations of the shocks are also displayed. The shock to the state variable `u` is denoted `e.u` and has standard deviation 2.32. The shock to the state variable `g` is denoted `e.g` and has standard deviation 0.61.

An example: A linear DSGE model

Writing down linearized DSGEs

The model in (4a)–(8a) is nonlinear. To use `dsge`, we first write the model in its corresponding linearized form, which we show below. Throughout this manual, we use lowercase letters to denote deviations of variables from the steady state. The linearized versions of (4a)–(8a) are

$$x_t = E_t x_{t+1} - (r_t - E_t \pi_{t+1} - g_t) \tag{4b}$$
$$\pi_t = \beta E_t \pi_{t+1} + \kappa x_t \tag{5b}$$
$$r_t = \frac{1}{\beta}\pi_t + u_t \tag{6b}$$
$$u_{t+1} = \rho_u u_t + \epsilon_{t+1} \tag{7b}$$
$$g_{t+1} = \rho_g g_t + \xi_{t+1} \tag{8b}$$

The new parameter κ is a function of the underlying parameter ϕ from (5a).

We have implicitly constrained some of the coefficients; for example, the coefficient on the interest rate in the output equation is constrained to -1.

Specifying the model to dsge

The `dsge` command to estimate the parameters in the system of (4b)–(8b) is

```
. dsge (p   = {beta}*F.p + {kappa}*x)              ///
       (x   = F.x - (r - F.p - g), unobserved)     ///
       (r   = (1/{beta})*p + u)                    ///
       (F.u = {rhou}*u, state)                     ///
       (F.g = {rhog}*g, state)
```

As with `dsgenl`, each equation is bound in parentheses. The equations look almost identical to the system in (4b)–(8b). Because a model has as many variables as it has equations, each variable will appear on the left-hand side of one and only one equation but can appear on the right-hand side of as many equations as we like. Equations may be specified in any order.

We identify types of variables differently with `dsge` than we did with `dsgenl`. Equation options that appear within the parentheses indicate which type of variable appears on the left-hand side of the equation. Equations specified without any options are equations for observed control variables. We add the `unobserved` option to specify that a left-hand-side variable is an unobserved control variable. We use the `state` option to identify state variables.

In our model, `p` and `r` are the observed control variables p_t and r_t, and their equations are specified without options. The output gap, x_t, is an unobserved control variable, so we specify the equation for `x` using the `unobserved` option.

`u` and `g` are the state variables u_t and g_t, and option `state` is included when we type their equations. Recall that state variables are fixed in the current period, so we specify how they evolve through time by modeling the one-period lead—hence, the `F.` on the left-hand side of each state equation. The state equations specify how the state variable evolves as a function of the current state variables and, possibly, the control variables.

The shocks ϵ_t and ξ_t enter the system through the state equations of their corresponding variable. By default, a shock is attached to each state equation. The number of shocks must be the same as the number of observed control variables in the model.

When we typed

```
. dsge ... (F.u = {rhou}*u, state) ...
```

the underlying equation is what we wrote in (7*b*). If a state variable is treated as deterministic in your model, then it will not have a shock. For example, capital accumulation is often treated as deterministic. To include an equation for a state variable without a shock, we would include the `noshock` option within the equation.

As with `dsgenl`, the operator `F.` can be applied to both state and control variables to denote the expected future value of a variable. For example, we can type `F.x` to specify x_{t+1}. Unlike `F.` used in other time-series applications, the value of x_{t+1} is not taken from the value of `x` at time $t+1$ in your data. Instead, it is an expectation of the value of `x` at time $t+1$ based on your model. `dsge` interprets equations with `F.` operators in the following manner. The equation

```
. dsge (p = {beta}*F.p + {kappa}*x) ...
```

is interpreted as

$$E_t\left(p_t - \beta p_{t+1} - \kappa x_t\right) = 0$$

which is equivalent to (4*b*) above. In a linear DSGE model, it is immaterial whether the expectation operator is thought of as applying to a variable, or to an equation, or indeed to the entire system of equations jointly.

The parameters we want to estimate are bound in curly braces.

For more details on the `dsge` syntax, see [DSGE] **Intro 2**.

Parameter estimation and interpretation of linear DSGEs

We estimate the parameters of the model in (4*b*)–(8*b*). These equations are much discussed in the monetary economics literature. Equation (4*b*) is known as the output-gap Euler equation. Equation (5*b*) is known as a New Keynesian Phillips Curve, and the parameter κ is known as the slope of the Phillips curve. In New Keynesian models, prices depend on output, and κ is a measure of that dependence. Equation (6*a*) is known as a Taylor rule, after Taylor (1993). The coefficient on inflation in a Taylor rule is a commonly discussed parameter. β has two roles in the model above. It relates current inflation deviations to expected future inflation deviations, and it relates interest rate deviations to inflation deviations.

```
. dsge (p = {beta}*F.p + {kappa}*x)
>       (x = F.x - (r - F.p - g), unobserved)
>       (r = (1/{beta})*p + u)
>       (F.u = {rhou}*u, state)
>       (F.g = {rhog}*g, state)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -13931.564
Iteration 1:  Log likelihood = -1301.5118  (backed up)
Iteration 2:  Log likelihood = -1039.6984  (backed up)
Iteration 3:  Log likelihood = -905.70867  (backed up)
Iteration 4:  Log likelihood = -842.76867  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -812.04209  (backed up)
Iteration 6:  Log likelihood = -787.29942
Iteration 7:  Log likelihood = -777.50219
Iteration 8:  Log likelihood = -768.84849
Iteration 9:  Log likelihood = -768.13605
Iteration 10: Log likelihood =  -768.0951
Iteration 11: Log likelihood = -768.09383
Iteration 12: Log likelihood = -768.09383
DSGE model
Sample: 1954q3 thru 2016q4                              Number of obs = 250
Log likelihood = -768.09383
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.5112878	.0757909	6.75	0.000	.3627403	.6598353
kappa	.1696301	.0475493	3.57	0.000	.0764353	.2628249
rhou	.6989185	.0449192	15.56	0.000	.6108785	.7869585
rhog	.9556407	.0181342	52.70	0.000	.9200984	.9911831
sd(e.u)	2.31759	.2988025			1.731948	2.903232
sd(e.g)	.614735	.0973277			.4239761	.8054938

Two of the parameters have structural interpretations. The parameter `kappa` is the slope of the Phillips curve. Theory predicts that this parameter will be positive, and indeed our estimate is positive.

The parameter `beta` is the inverse of the coefficient on inflation in the interest rate equation. We can obtain an estimate of $1/\beta$, which is interpreted as the degree to which the central bank responds to movements in inflation, by using `nlcom`.

```
. nlcom 1/_b[beta]
       _nl_1: 1/_b[beta]
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
_nl_1	1.955846	.2899255	6.75	0.000	1.387602	2.524089

A typical value for $1/\beta$ in the literature is 1.5. Our point estimate is around 2.

Postestimation

Several tools are available to help you analyze your model after you have estimated its parameters. These tools are available after using either `dsgenl` or `dsge`. Below, we continue with the model we fit using `dsge`. The results would be the same if you used these commands after `dsgenl`.

Policy and transition matrices

The matrix of parameters in the state-space form that specifies how the state variables affect the control variables is known as the policy matrix. Each policy matrix parameter is the effect of a one-unit shock to a state variable on a control variable.

We use `estat policy` to view these results.

```
. estat policy
Policy matrix
```

		Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
p							
	u	-.4170858	.0389324	-10.71	0.000	-.4933918	-.3407798
	g	.8818832	.2330568	3.78	0.000	.4251002	1.338666
x							
	u	-1.580149	.3926325	-4.02	0.000	-2.349694	-.8106032
	g	2.658658	.9045254	2.94	0.003	.8858213	4.431496
r							
	u	.1842446	.0567979	3.24	0.001	.0729228	.2955665
	g	1.724827	.2210255	7.80	0.000	1.291625	2.158029

Results are listed equation by equation. The first block is the policy equation for inflation `p` and writes it as a function of the state variables alone. A unit shock to the state `u` reduces inflation by an estimated 0.417, and a unit shock to `g` raises inflation by an estimated 0.882.

The matrix of parameters that specifies the dynamic process for the state variables is known as the state transition matrix. The state transition equation relates the future values of the state variables to their values in the current period. Each state transition matrix parameter is the effect of a one-unit shock to a state variable on its one-period-ahead mean.

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

	Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
F.u						
u	.6989185	.0449192	15.56	0.000	.6108785	.7869585
g	1.11e-16	.	.	.	.	.
F.g						
u	0	(omitted)				
g	.9556407	.0181342	52.70	0.000	.9200984	.9911831

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

Both state variables are modeled as autoregressive processes, so the results in `estat transition` repeat the estimates of `rhou` and `rhog` from the `dsge` output. In this case, the other entries in the state transition matrix are 0 or differ from 0 only because of lack of numerical precision. In more complicated models, such as those in which a state equation depends on a control variable, the state transition matrix will contain new information about that state variable.

Impulse responses

The state-space form allows us to trace the path of a control or state in response to a shock to a state. This path is known as an impulse–response function (IRF). `irf` after `dsge` or `dsgenl` estimates IRFs, and it puts the named set of estimates into an `.irf` file, whose results can be displayed using `irf graph` or `irf table`.

To graph the IRF, we first create the `nkirf.irf` file and set it as the active `.irf` file using the `irf set` command.

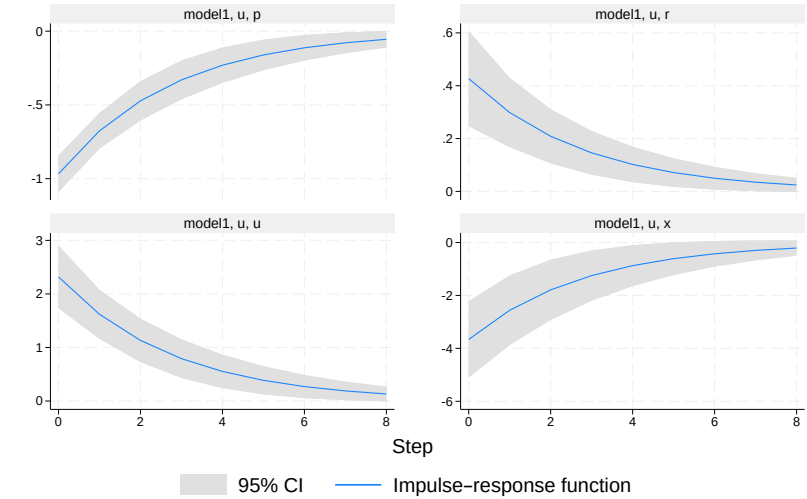
```
. irf set nkirf.irf
(file nkirf.irf created)
(file nkirf.irf now active)
```

Next, we use `irf create` to estimate a complete set of impulse responses based on our `dsge` command. A complete set of impulse responses is an impulse to each shock and the response of each state and control variable to that impulse. For the model in this example, `irf create` generates an impulse to `e.u` and `e.g`, then stores the response to that impulse on `p`, `x`, `r`, `g`, and `u`. The results are stored in the `nkirf.irf` file.

```
. irf create model1
(file nkirf.irf updated)
```

We now use `irf graph` to plot the impulse responses. The `impulse()` and `response()` options control which impulse and which responses are chosen. To view the response of `p`, `x`, `r`, and `u` to a shock to `u`, we type

```
. irf graph irf, impulse(u) response(x p r u) byopts(yrescale)
```



Graphs by irfname, impulse variable, and response variable

The state variable `u` models movements in the interest rate that occur for reasons other than the feedback between inflation and the interest rate. A shock to `u` is effectively a surprise increase in the interest rate, and the IRF traces out how this shock causes temporary decreases to inflation (top-left graph) and to the output gap (bottom-right graph).

Forecasts

The `forecast` suite of commands produces dynamic forecasts from the fitted model.

We first store the `dsge` estimation results.

```
. estimates store dsge_est
```

We use `tsappend()` to extend the dataset by 3 years, or 12 quarters.

```
. tsappend, add(12)
```

To set up a forecast, we perform three steps. First, we type `forecast create` to initialize a new forecasting model, which we name `dsgemodel`.

```
. forecast create dsgemodel
Forecast model dsgemodel started.
```

Next, we add the estimates from `dsge` to the forecasting model using `forecast estimates`.

```
. forecast estimates dsge_est
Added estimation results from dsge.
Forecast model dsgemodel now contains 2 endogenous variables.
```

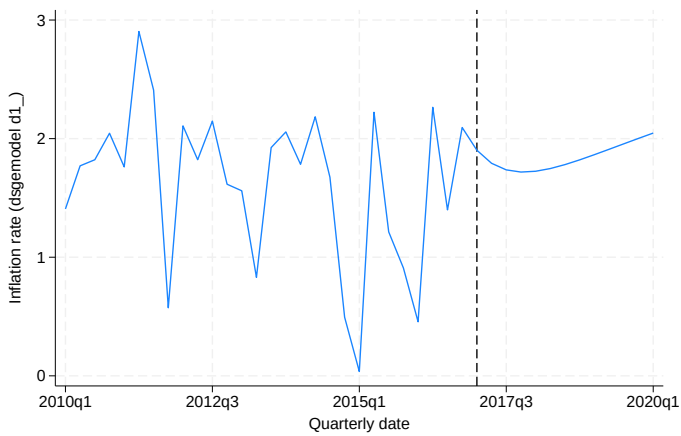
This command adds the estimates stored in `dsge_est` to the model `dsgemodel`. Finally, we produce dynamic forecasts beginning in the first quarter of 2017 using `forecast solve`. The `prefix(d1_)` option specifies the `d1_` prefix that will be given to the variables created by `forecast`. We also request that dynamic forecasts begin in the first quarter of 2017 with the `begin(tq(2017q1))` option.

```
. forecast solve, prefix(d1_) begin(tq(2017q1))
Computing dynamic forecasts for model dsgemodel.
------------------------------------------------------------

Starting period:  2017q1
Ending period:    2020q1
Forecast prefix:  d1_

2017q1:  ..............
2017q2:  ..............
2017q3:  ..............
2017q4:  ..............
2018q1:  ..............
2018q2:  ..............
2018q3:  ..............
2018q4:  ..............
2019q1:  ..............
2019q2:  ..............
2019q3:  ..............
2019q4:  ..............
2020q1:  ..............

Forecast 2 variables spanning 13 periods.
------------------------------------------------------------
```

The dynamic forecast begins in the first quarter of 2017, so all forecasts are out of sample.

We can graph the forecast for inflation `d1_p` using `tsline`.

```
. tsline d1_p if tin(2010q1, 2021q1), tline(2017q1)
```



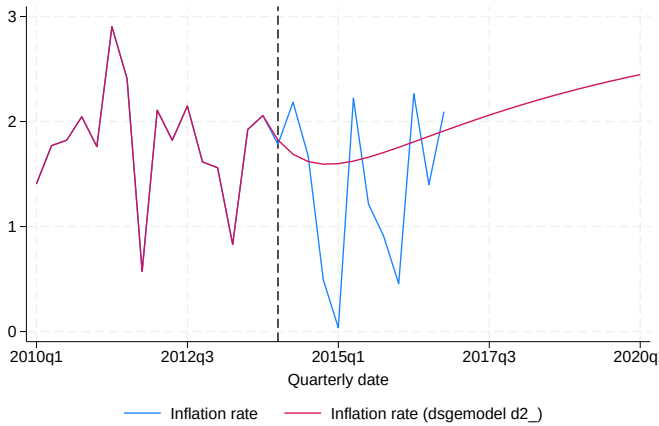
The model forecasts that inflation will smoothly return to its long-run value, the sample mean.

We can also begin the forecast during a time period for which observations are available.

Specifying the `begin(tq(2014q1))` option produces dynamic forecasts beginning in the first quarter of 2014, so we can compare the forecast for 2014–2016 with the actual observations over that period.

```
. forecast solve, prefix(d2_) begin(tq(2014q1))

Computing dynamic forecasts for model dsgemodel.
------------------------------------------------------------

Starting period:  2014q1
Ending period:    2020q1
Forecast prefix:  d2_

2014q1:  ..............
2014q2:  ..............
2014q3:  ..............
2014q4:  ..............
2015q1:  ..............
2015q2:  ..............
2015q3:  ..............
2015q4:  ..............
2016q1:  ..............
2016q2:  ..............
2016q3:  ..............
2016q4:  ..............
2017q1:  ..............
2017q2:  ..............
2017q3:  ..............
2017q4:  ..............
2018q1:  ..............
2018q2:  ..............
2018q3:  ..............
2018q4:  ..............
2019q1:  ..............
2019q2:  ..............
2019q3:  ..............
2019q4:  ..............
2020q1:  ..............

Forecast 2 variables spanning 25 periods.
------------------------------------------------------------

. tsline p d2_p if tin(2010q1, 2021q1), tline(2014q1)
```

We plot both the observed inflation and the forecast.



The forecast captures the general upward trend in inflation from 2014–2016, but it does not predict the variation in inflation around the upward trend.

Structural and reduced forms of DSGE models

Now that we have worked an example, we show how it fits in the more general formulation of DSGE models. The model in (4*a*)–(8a) is an example of a nonlinear DSGE model. In general, a nonlinear DSGE model can be expressed as

$$E_t\left\{\mathbf{f}(\mathbf{x}_{t+1},\mathbf{y}_{t+1},\mathbf{x}_t,\mathbf{y}_t,\boldsymbol{\theta})\right\}=\mathbf{0} \tag{9}$$

where $\mathbf{f}$ is a vector of equations, $\mathbf{x}_t$ is a vector of state variables, $\mathbf{y}_t$ is a vector of control variables, and $\boldsymbol{\theta}$ denotes the vector of structural parameters. To solve a DSGE model means to write it in state-space form. In general, the state-space form of a nonlinear DSGE model is nonlinear. `dsgenl` takes a linear approximation to the model equations to solve for a linear approximation to the state-space form.

The model in (4*b*)–(8*b*) is an example of a linearized DSGE model. In general, a linearized DSGE model can be expressed as

$$\mathbf{A}_0\mathbf{y}_t=\mathbf{A}_1E_t(\mathbf{y}_{t+1})+\mathbf{A}_2\mathbf{y}_t+\mathbf{A}_3\mathbf{x}_t \tag{10}$$
$$\mathbf{B}_0\mathbf{x}_{t+1}=\mathbf{B}_1E_t(\mathbf{y}_{t+1})+\mathbf{B}_2\mathbf{y}_t+\mathbf{B}_3\mathbf{x}_t+\mathbf{C}\boldsymbol{\epsilon}_{t+1} \tag{11}$$

where as before $\mathbf{y}_t$ is a vector of control variables, $\mathbf{x}_t$ is a vector of state variables, and $\boldsymbol{\epsilon}_t$ is a vector of shocks. $\mathbf{A}_0$ through $\mathbf{A}_3$ and $\mathbf{B}_0$ through $\mathbf{B}_3$ are matrices of parameters. We require that $\mathbf{A}_0$ and $\mathbf{B}_0$ be diagonal matrices. The entries in all of these matrices are functions of the structural parameters, which we denote by vector $\boldsymbol{\theta}$. Economic theory places restrictions on the entries in the matrices. $\mathbf{C}$ is a selection matrix that determines which state variables are subject to shocks.

The reduced form of a DSGE model expresses the control variables as functions of the state variables alone and specifies how state variables evolve over time. The reduced form is a state-space model. The state-space form of the model is given by

$$\mathbf{y}_t=\mathbf{G}\mathbf{x}_t \tag{12}$$
$$\mathbf{x}_{t+1}=\mathbf{H}\mathbf{x}_t+\mathbf{M}\boldsymbol{\epsilon}_{t+1} \tag{13}$$

where $\mathbf{y}_t$ is a vector of control variables, $\mathbf{x}_t$ is a vector of state variables, and $\boldsymbol{\epsilon}_t$ is a vector of shocks. $\mathbf{G}$ is the policy matrix, and $\mathbf{H}$ is the state transition matrix. $\mathbf{M}$ is diagonal and contains the standard deviations of the shocks.

$\mathbf{y}_t$ is partitioned into observed and unobserved controls, $\mathbf{y}_t = (\mathbf{y}_{1,t}, \mathbf{y}_{2,t})$. The observed control variables are related to the control variables by the equation

$$\mathbf{y}_{1,t} = \mathbf{D}\mathbf{y}_t$$

where $\mathbf{D}$ is a selection matrix. Only observed control variables play a role in estimation. The number of observed control variables must be the same as the number of state equations that include shocks.

You specify a model of the form (9) to `dsgenl` and a model of the form (10)–(11) to `dsge`. Many models require some manipulation to fit into the structure in (9) and (10); see [DSGE] **Intro 4** for details. Postestimation commands `estat policy` and `estat transition` will display the policy and transition matrices in (12) and (13), respectively.

References

Canova, F. 2007. *Methods for Applied Macroeconomic Research*. Princeton, NJ: Princeton University Press.

DeJong, D. N., and C. Dave. 2011. *Structural Macroeconometrics*. 2nd ed. Princeton, NJ: Princeton University Press.

Fernández-Villaverde, J., J. F. Rubio-Ramírez, and F. Schorfheide. 2016. Solution and estimation methods for DSGE models. In Vol. 2A of *Handbook of Macroeconomics*, ed. J. B. Taylor and H. Uhlig, chap. 9, 527–724. Amsterdam: North-Holland. https://doi.org/10.1016/bs.hesmac.2016.03.006.

Ljungqvist, L., and T. J. Sargent. 2018. *Recursive Macroeconomic Theory*. 4th ed. Cambridge, MA: MIT Press.

Taylor, J. B. 1993. Discretion versus policy rules in practice. *Carnegie-Rochester Conference Series on Public Policy* 39: 195–214. https://doi.org/10.1016/0167-2231(93)90009-L.

Woodford, M. 2003. *Interest and Prices: Foundations of a Theory of Monetary Policy*. Princeton, NJ: Princeton University Press.

Also see

[DSGE] **dsge** — Linear dynamic stochastic general equilibrium models

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

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

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

[DSGE] **Intro 9** — Bayesian estimation

[TS] **forecast** — Econometric model forecasting

[TS] **irf** — Create and analyze IRFs, dynamic-multiplier functions, and FEVDs

[TS] **sspace** — State-space models

Title

Intro 2 — Learning the syntax

Description Remarks and examples Also see

Description

In this introduction, we demonstrate how to specify a DSGE model using the `dsge` and `dsgenl` commands. We focus on two unique aspects of DSGEs that must be considered when writing the syntax—writing the system of equations and identifying each type of variable within those equations.

Remarks and examples

Remarks are presented under the following headings:

Introduction

If you have not read [DSGE] **Intro 1**, we recommend that you read it first. In particular, see *Structural and reduced forms of DSGE models* in [DSGE] **Intro 1**, where we discuss the structural forms of a DSGE model that are required by `dsge` and `dsgenl`. Below, we assume that your model has this structural form, and we discuss the syntax for specifying such a model with the `dsge` and `dsgenl` commands.

`dsge` estimates the parameters of linear DSGE models. `dsgenl` estimates the parameters of nonlinear DSGE models. The two commands have elements in common. Both center around a list of equations and a collection of variables. The two commands differ somewhat in how they expect you to specify equations and variables. Below, we first describe the syntax for `dsge` and then for `dsgenl`.

Syntax for linear DSGE models

Preview of dsge syntax

As an example, if we wanted to fit the DSGE model

$$
\begin{aligned}
p_t &= \beta E_t(p_{t+1}) + \kappa y_t \\
y_t &= E_t(y_{t+1}) - \{r_t - E_t(p_{t+1}) - \rho z_t\} \\
\beta r_t &= p_t + \beta u_t \\
z_{t+1} &= \rho z_t + \epsilon_{t+1} \\
u_{t+1} &= \delta u_t + \xi_{t+1}
\end{aligned}
$$

we would type

```
. dsge (p        = {beta}*F.p + {kappa}*y)                          ///
       (y        = F.y -(r - F.p - {rho}*z), unobserved)            ///
       ({beta}*r = p + {beta}*u)                                    ///
       (F.z      = {rho}*z, state)                                  ///
       (F.u      = {delta}*u, state)
```

The syntax looks a lot like the equations. In what follows, we discuss each element of the `dsge` syntax that you will need to specify the equations of your DSGE model.

In [DSGE] **Intro 1**, we told you that three types of variables—control variables, state variables, and shocks—appear in DSGE models. We will demonstrate how to specify equations involving each of these types of variables. We will also show you how to specify the required types of equations that are linear in these variables and nonlinear in the parameters.

Specifying the system of linear equations

For a linear DSGE model, we specify one equation for each control variable and one equation for each state variable. The equations are bound by parentheses. Therefore, the basic structure of the `dsge` command is

```
. dsge (eq1) (eq2) ..., ...
```

where *eq1* and *eq2* will be replaced with the syntax representing one of the equations in our model. We use ... to indicate that we can include more than two equations as well as options.

The basic form of an equation within the parentheses is

term = *term* [+ *term* [+ *term* [...]]]

where each *term* includes a variable name. The variable name may be preceded by a parameter or nonlinear combination of parameters. For instance, a valid equation might look something like

```
(y = {kappa}*z)
```

or

```
(y = {kappa}*z + {kappa}*{beta}*x)
```

or

```
(1/{beta}*y = {gamma}*z + x)
```

We will explain this later. For now, simply note that these equations are written in dsge syntax much like we would write the math, but the parameters we want to estimate are offset with curly braces, {}. If you have used any other Stata commands that work with substitutable expressions, you may recognize this notation. In fact, dsge uses a special form of substitutable expressions.

Before we discuss how to use dsge's substitutable expressions, we will focus on how to specify each type of variable.

Control variables

Control variables can be modeled as a function of other control variables, expectations of the future value of control variables, and state variables. The equations for control variables do not include shocks.

We continue from the basic dsge command in the previous section, which was

```
. dsge (eq1) (eq2) ..., ...
```

If we have an observed control variable y, and for *eq1*, we want to specify that y_t be modeled as a function of z_t and x_t, we can type this equation in our dsge command as

```
. dsge (y = pexp*z + pexp*x)          ///
  (eq2)                               ///
  ..., ...
```

where *pexp* is a possibly nonlinear expression of parameters in each term.

Control variables can be observed or unobserved. Because we did not include any options within this set of parentheses, y is assumed to be an observed control variable. If y were unobserved, we would add the unobserved option as follows:

```
(y = pexp*z + pexp*x, unobserved)
```

Note that each control variable in the model must be included on the left-hand side of one, and only one, equation.

State variables and shocks

To model a state variable, we specify an equation with the one-period lead of that state variable on the left-hand side. On the right-hand side of the equation, we can include state variables, control variables, expectations of the future value of control variables, and shocks.

When we specify an equation for a state variable, we include the state option within the parentheses defining the equation.

Continuing with the syntax above, let's suppose *eq2* is an equation for a state variable x, and we model x_{t+1} as a function of x_t. We expand our dsge command to

```
. dsge (y = pexp*z + pexp*x)          ///
  (F.x = pexp*x, state)               ///
  ..., ...
```

We used the F. lead operator to specify the one-period lead of x as F.x; see [U] **11.4.4 Time-series varlists**. However, notice that the full list of time-series operators is not available here. We can specify only equations for one-period leads of state variables; we could not replace F.x with F2.x in the equation above. Note that this restriction does not limit the types of models we can fit; see [DSGE] **Intro 4c**.

By default, the equation for a state variable includes an unobserved shock. However, equations for state variables are not required to include a shock. Within the system of equations, the number of shocks should be equal to the number of observed control variables. If we did not wish to include a shock in the equation for x_{t+1}, we could add the `noshock` option,

```
(F.x = pexp*x, state noshock)
```

Note that the one-period lead of each state variable must be included on the left-hand side of one, and only one, equation.

Expectations of future values of control variables

Expectations of the one-period lead of control variables can appear in equations for both control and state variables. Mathematically, we write these expectations as $E_t(\cdot)$. For instance, we write the expectation of y in time $t+1$ as $E_t(y_{t+1})$. In `dsge`, we write this expectation as `F.y`. Any time `dsge` sees the `F.` operator applied to a control variable, it interprets this as the expectation of that variable at time $t+1$.

If we model y_t as a function of $E_t(y_{t+1})$ in addition to z_t and x_t, we can expand our previous `dsge` command to

```
. dsge (y = pexp*F.y + pexp*z + pexp*x)          ///
  (F.x = pexp*x, state)                          ///
  ..., ...
```

Expectations are strictly for one-period leads of control variables. You cannot, for instance, use `F2.y` to include the expectation of y in time $t+2$ in the model. This does not prevent you, however, from including such terms in your model. See [DSGE] **Intro 4d** for details of fitting models, including expectations of control variables more than one period in the future.

Specifying parameters using dsge's substitutable expressions

At this point, we know how to specify each type of variable that may arise in our DSGE model. We now turn to specifying the parameters that we want to estimate.

Recall that the basic form of an equation is

```
term = term [ + term [ + term [...]]]
```

where each *term* includes one variable name. The variable name may be preceded by a parameter or a nonlinear combination of parameters. For *term*s on the right-hand side of the equation, variable names can also be followed by a parameter specification.

We specify the *term*s using a special type of substitutable expressions that we call scalar substitutable expressions. In scalar substitutable expressions, parameters are enclosed in curly braces, `{}`, and may enter the model either linearly or nonlinearly. The restriction that each term includes only one variable implies that the variables must enter the equation linearly with these scalar substitutable expressions.

If our model for y in the equation above is

$$y_t = \beta E_t(y_{t+1}) + \kappa z_t + \gamma x_t$$

we could extend our previous `dsge` command as follows:

```
. dsge (y = {beta}*F.y + {kappa}*z + {gamma}*x)   ///
  (F.x = pexp*x, state)                           ///
  ..., ...
```

However, the equation for y need not be linear in the parameters. If instead we wanted to model y as

$$y_t = (1/\beta)E_t(y_{t+1}) + \kappa z_t + (\gamma/\beta)x_t$$

we would change our command to

```
. dsge (y = 1/{beta}*F.y + {kappa}*z + ({gamma}/{beta})*x)  ///
  (F.x = exp*x, state)                                      ///
  ..., ...
```

We can even include parameters on the left-hand side of the equation. For instance, we could model y as

$$(1/\beta)y_t = E_t(y_{t+1}) + \kappa z_t + (\gamma/\beta)x_t$$

and change our `dsge` command to

```
. dsge ((1/{beta})*y = F.y + {kappa}*z + ({gamma}/{beta})*x)  ///
  (F.x = exp*x, state)                                       ///
  ..., ...
```

Finally, there is an extension to the basic form of the equation that we should mention. Sometimes, it is more convenient to write an equation so that a scalar value or a parameter or a nonlinear combination of parameters is multiplied by a linear combination of terms. The `dsge` command allows this. For instance, instead of

$$y_t = (1/\beta)E_t(y_{t+1}) + (\gamma/\beta)x_t + \kappa z_t$$

we can write

$$y_t = (1/\beta)\{E_t(y_{t+1}) + \gamma x_t\} + \kappa z_t$$

Similarly, we could write this equation in the `dsge` command as

```
(y = (1/{beta}) * (F.y + {gamma}*x) + {kappa}*z)
```

Now you know the general syntax of `dsge` and are ready to fit your own DSGE model with `dsge`. For examples of fitting classic DSGE models using this syntax, see [DSGE] **Intro 3a**, [DSGE] **Intro 3b**, and [DSGE] **Intro 3c**. For more details on the `dsge` syntax, see [DSGE] **dsge**.

Syntax for nonlinear DSGE models

The `dsgenl` syntax builds on the `dsge` syntax. However, because of the additional complexity of nonlinear models, the syntax does differ. As we discuss the `dsgenl` syntax below, we will highlight similarities and differences from `dsge`.

Preview of dsgenl syntax

Suppose we wanted to estimate the parameters of the nonlinear DSGE model,

$$
\begin{aligned}
R_t &= \alpha \frac{Y_t}{K_t} \\
1 &= \beta E_t \left\{ \left(\frac{C_t}{C_{t+1}} \right) (1 + R_{t+1} - \delta) \right\} \\
Y_t &= Z_t K_t^{\alpha} \\
K_{t+1} &= Y_t - C_t + (1 - \delta) K_t \\
\ln(Z_{t+1}) &= \rho \ln(Z_t) + e_{t+1}
\end{aligned}
$$

To estimate the parameters of this model, we would type

```
. dsgenl (r = {alpha}*y/k)                          ///
         (1 = {beta}*(c/F.c)*(1 + F.r - {delta})) ///
         (y = z*k^({alpha}))                        ///
         (F.k = y - c + (1-{delta})*k)              ///
         (ln(F.z) = {rho}*ln(z)) ,                  ///
         observed(y) unobserved(c r) exostate(z) endostate(k)
```

As with `dsge`, the `dsgenl` syntax looks a lot like the equations. Variables are written as normal, and parameters are written bound by curly braces. We first focus on two syntax elements: the system of equations that specifies the interrelations among variables and the options that specify how each variable is treated.

Specifying the system of nonlinear equations

For a nonlinear DSGE model, we specify a system of equations that jointly determines the structure of the model. There must be as many equations as there are control and state variables. The equations are bound by parentheses. Therefore, the basic structure of the `dsgenl` command is

```
. dsgenl (eq1) (eq2) ..., ...
```

where *eq1* and *eq2* will be replaced with the syntax representing one of the equations in our model. We use ... to indicate that we can include more than two equations as well as options.

The equations for a nonlinear model fit with `dsgenl` take on a much more flexible form than the equations we described for a linear model fit by `dsge`. For `dsgenl`, the equation within the parentheses may include a nonlinear substitutable expression on both sides of the equal sign. An example of such an expression is the first equation in the model above, which reads

```
(r = {alpha}*y/k)
```

The equations written in `dsgenl`, like those in `dsge`, use curly braces to identify the parameters such as `alpha` that are to be estimated.

This equation could have been expressed in other ways. As written above, it reads like an equation for `r`. We could just as well have written it as

```
(k = {alpha}*y/r)
```

or

```
(r*k/y = {alpha})
```

or even

```
(r*k/y - {alpha} = 0)
```

All four of the above Stata equations specify the same mathematical equation, and it does not matter which parameterization you choose.

The second equation in the model above reads

```
(1 = {beta}*(c/F.c)*(1 + F.r - {delta}))
```

We could have written it as

```
(1/c = {beta}*(1/F.c)*(1 + F.r - {delta}))
```

or

```
(1 = {beta}*(F.c/c)^(-1)*(1 + F.r - {delta}))
```

The three equations just listed specify the same mathematical equation, and it does not matter which parameterization you choose.

In the second equation of the model, the `F.` lead operator denotes the future value of a variable, just as it did in `dsge`. More information on how `dsgenl` treats the `F.` operator is below, under *Expectations in nonlinear models*.

State and control variables

So far, we have discussed only the specification of model equations. We have seen that you can specify the model equations in nearly any form. The price of that freedom is that you must specify a set of options—`exostate()`, `endostate()`, `observed()`, and `unobserved()`—that tell Stata what role each variable plays in the model.

Recall that a DSGE model has control and state variables. Each control variable may be observed or unobserved, and each state variable may be subject to shocks. Hence, each variable fits into one of four types: observed controls, unobserved controls, a state variable that is not subject to a shock, and a state variable that is subject to a shock. The above model has variables of all four types.

Using the jargon common in macroeconomics, we call a state variable that is subject to a shock an exogenous state variable. We call a state variable that is not subject to a shock an endogenous state variable. You must have exactly as many exogenous state variables (and thus, as many shocks) as you have observed control variables.

In the model above, the option list

```
. dsgenl ..., ... observed(y) unobserved(c r) exostate(z) endostate(k) ...
```

specifies that `c`, `r`, and `y` are control variables; of those, `y` is observed. The option list specifies `k` and `z` as state variables, of which `z` is subject to shocks.

Expectations in nonlinear models

In *Expectations of future values of control variables*, we saw that terms in models fit with `dsge` can include the expectation of the one-period lead of a variable. Such expectations can also be included in models fit with `dsgenl`. Because the equations in `dsgenl` are more flexible, we need to think of the expectations in a more general way, as we describe below.

Expectations of the one-period lead of variables can appear in any equation. Mathematically, we write these expectations as $E_t(\cdot)$. For instance, we write the expectation of y in time $t+1$ as $E_t(y_{t+1})$. As we did in `dsge`, we write this expectation as `F.y`.

In a nonlinear model, however, the expectation may not be applied directly to one variable. We might have

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

where the expectation is written as if it is applied to everything in the curly brackets. Really, the expectation can be thought of as being applied to the entire equation. This means we can use the `F.` lead operator and write the equation involving the expectation above as

```
. dsgenl (1 = {beta}*(c/F.c)*(1 + F.r - {delta})) ..., ...
```

in `dsgenl`, and it will be interpreted as

$$E_t \left\{ 1 - \beta \left(\frac{C_t}{C_{t+1}} \right) (1 + R_{t+1} - \delta) \right\} = 0$$

The upshot is that you can specify expected future values by using the `F.x` notation, and both `dsgenl` and `dsge` will take the appropriate action.

Now you have seen all elements of the general syntax `dsgenl` and are ready to fit your own model. For examples of fitting classic nonlinear DSGE models, see [DSGE] **Intro 3d**, [DSGE] **Intro 3e**, and [DSGE] **Intro 3f**. For more details on the `dsgenl` syntax, see [DSGE] **dsgenl**.

Also see

[DSGE] **dsge** — Linear dynamic stochastic general equilibrium models

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

[DSGE] **Intro 3** — Classic DSGE examples

[DSGE] **Intro 4** — Writing a DSGE in a solvable form

Title

Intro 3 — Classic DSGE examples

Description Remarks and examples Also see

Description

In this entry, we present several classic DSGE examples. These include linear and nonlinear versions of a New Keynesian model, linear and nonlinear versions of a New Classical model (Real Business Cycle model), a linear financial frictions model, and a nonlinear stochastic growth model.

Remarks and examples

In [DSGE] **Intro 3a**, [DSGE] **Intro 3b**, and [DSGE] **Intro 3c**, we fit simple variants of common linearized DSGE models. In [DSGE] **Intro 3d**, [DSGE] **Intro 3e**, and [DSGE] **Intro 3f**, we fit simple variants of common nonlinear DSGE models. Through these examples, we demonstrate model solution, estimation, and interpretation.

[DSGE] **Intro 3a** demonstrates how to fit a New Keynesian model. In this example, we interpret structural parameters, policy matrix parameters, and state transition matrix parameters. We also predict values of both observed control variables and unobserved states.

[DSGE] **Intro 3b** illustrates how to solve a New Classical model and plot the IRFs to compare the model's theoretical predictions under different sets of parameter values.

[DSGE] **Intro 3c** fits a financial frictions model. In this example, we also estimate parameters of the policy matrix and evaluate the IRFs.

[DSGE] **Intro 3d** revisits the New Keynesian model. In this example, we estimate the parameters of the nonlinear model, interpret structural parameters, and explain partial identification by parameter restrictions in a DSGE model.

[DSGE] **Intro 3e** revisits the New Classical model. We fix some parameters and estimate others, explain the effect of fixing parameters on postestimation statistics, and explain how to compare models across different parameter settings.

[DSGE] **Intro 3f** demonstrates how to solve a nonlinear stochastic growth model. We explain how `dsgenl` takes an approximation of the model and how to interpret the steady state and approximations to the policy and transition equations. This example describes some of the technical differences between linear and log-linear approximations to nonlinear DSGE models.

Also see

[DSGE] **Intro 4** — Writing a DSGE in a solvable form

[DSGE] **dsge** — Linear dynamic stochastic general equilibrium models

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

Title

Intro 3a — New Keynesian model

Description Remarks and examples Reference Also see

Description

This introduction estimates and interprets the parameters of a simple New Keynesian model. In this entry, we demonstrate how to constrain parameters in the model and how to interpret structural parameters, policy matrix parameters, and state transition matrix parameters. We also predict values of both observed control variables and unobserved states. For Bayesian analysis of this model, see [DSGE] **Intro 9a**.

Remarks and examples

Remarks are presented under the following headings:

The model

Equations (1)–(5) specify a canonical New Keynesian model of inflation p_t, the output gap x_t, and the interest rate r_t. These are the linearized equations; the model's nonlinear equations are similar to those in *Writing down nonlinear DSGEs* in [DSGE] **Intro 1**.

$$p_t = \beta E_t(p_{t+1}) + \kappa x_t \tag{1}$$
$$x_t = E_t(x_{t+1}) - \{r_t - E_t(p_{t+1}) - g_t\} \tag{2}$$
$$r_t = \psi p_t + u_t \tag{3}$$
$$u_{t+1} = \rho_u u_t + \epsilon_{t+1} \tag{4}$$
$$g_{t+1} = \rho_g g_t + \xi_{t+1} \tag{5}$$

Equation (1) specifies inflation as a linear combination of expected future inflation and the output gap. Equation (2) specifies the output gap as a linear combination of the expected future output gap, the real interest rate, and a state variable g_t. Equation (3) specifies the interest rate as a linear combination of inflation and a state variable u_t. The state variables are modeled as first-order autoregressive processes. The state variable u_t is the deviation of r_t from its equilibrium value of ψp_t. The state variable g_t is also the deviation of x_t from its equilibrium value.

Three of the parameters have structural interpretation. The parameter κ is known as the slope of the Phillips curve and is predicted to be positive. The parameter β is the discount factor that represents the degree to which agents discount the future relative to the current period. The parameter ψ measures the degree to which interest rates react to movements in inflation.

Parameter estimation

Not all model parameters are identified. We constrain β to be 0.96, a common value in the literature. The remaining parameters are identified.

```
. use https://www.stata-press.com/data/r18/usmacro2
(Federal Reserve Economic Data - St. Louis Fed, 2017-01-15)
. constraint 1 _b[beta]=0.96

. dsge (p = {beta}*F.p + {kappa}*x)
>      (x = F.x - (r - F.p - g), unobserved)
>      (r = {psi}*p + u)
>      (F.u = {rhou}*u, state)
>      (F.g = {rhog}*g, state),
>     from(psi=1.5)  constraint(1)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -5736.4646
Iteration 1:  Log likelihood = -1190.0604  (backed up)
Iteration 2:  Log likelihood = -960.00953  (backed up)
Iteration 3:  Log likelihood = -928.79225  (backed up)
Iteration 4:  Log likelihood = -842.40806  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -810.92149  (backed up)
Iteration 6:  Log likelihood = -765.73779  (not concave)
Iteration 7:  Log likelihood = -759.67265
Iteration 8:  Log likelihood = -754.17723
Iteration 9:  Log likelihood = -753.59661
Iteration 10: Log likelihood = -753.57155
Iteration 11: Log likelihood = -753.57131
Iteration 12: Log likelihood = -753.57131
DSGE model
Sample: 1955q1 thru 2015q4                      Number of obs = 244
Log likelihood = -753.57131
 ( 1)  [/structural]beta = .96
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.96	(constrained)				
kappa	.0849631	.0287693	2.95	0.003	.0285763	.14135
psi	1.943004	.2957869	6.57	0.000	1.363272	2.522735
rhou	.7005483	.0452604	15.48	0.000	.6118397	.789257
rhog	.9545257	.0186424	51.20	0.000	.9179873	.9910641
sd(e.u)	2.318204	.3047434			1.720918	2.91549
sd(e.g)	.5689891	.0982975			.3763296	.7616486

The slope of the Phillips curve, `kappa`, is estimated to be positive. The coefficient on inflation in the interest rate equation is estimated to be almost 1.94, meaning that interest rates are expected to rise almost two for one with increases in inflation.

Policy and transition matrices

Elements of the policy matrix represent the response of a control variable to a one-unit increase in a state variable.

```
. estat policy
Policy matrix
```

		Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
p							
	u	-.4172521	.0393609	-10.60	0.000	-.494398	-.3401061
	g	.9678177	.2777452	3.48	0.000	.4234472	1.512188
x							
	u	-1.608216	.405054	-3.97	0.000	-2.402107	-.8143245
	g	.9529203	.4813931	1.98	0.048	.0094071	1.896433
r							
	u	.1892776	.059166	3.20	0.001	.0733143	.3052409
	g	1.880474	.2616	7.19	0.000	1.367747	2.3932

An increase in `u` decreases the extent to which the inflation is above its short-run equilibrium value. This change decreases the output gap and increases interest rate.

An increase in `g` increases the extent to which the inflation is above its short-run equilibrium value. This change also increases output gap and interest rates.

Because the states are uncorrelated with each other in this example, the elements of the state transition matrix are just the persistence parameters in the model.

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

		Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
F.u							
	u	.7005483	.0452604	15.48	0.000	.6118397	.789257
	g	3.33e-16	.	.	.	.	.
F.g							
	u	0	(omitted)				
	g	.9545257	.0186424	51.20	0.000	.9179873	.9910641

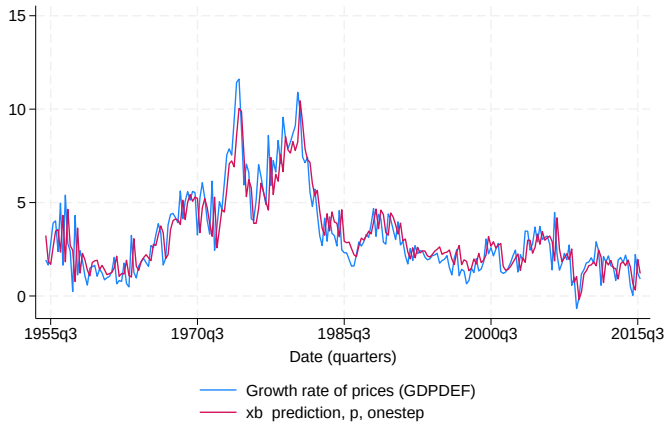
```
Note: Standard errors reported as missing for constrained transition matrix
      values.
```

See [DSGE] **Intro 4g** for an example in which the state variables depend on each other. The states can also depend on each other when a state variable is specified to depend on control variables, as in [DSGE] **Intro 3b**, or when the state vector includes lagged control variables, as in [DSGE] **Intro 4a**.

One-step-ahead predictions

Predictions after `dsge` depend on the estimated state-space parameters. Below, we obtain one-step-ahead predictions for each of the two observed control variables in the model, and we graph the actual and predicted inflation rates.

```
. predict dep*
(option xb assumed; fitted values)
. tsline p dep1, legend(col(1))
```



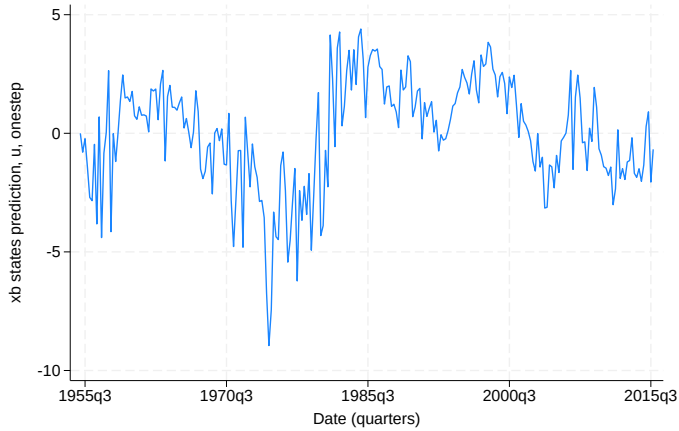
The graph shows that the one-step-ahead predictions closely follow realized inflation.

Estimating an unobserved state

The observed control variables are driven by two unobserved state variables. We can use `predict` with the `state` option to estimate the state variables.

Here we estimate the unobserved state u and plot it.

```
. predict state1, state
. tsline state1
```



The loose monetary policy in the mid-1970s, where predicted values of u_t are negative, and the subsequent Volcker contraction are apparent in this plot.

Reference

Schenck, D. 2017. Estimating the parameters of DSGE models. *The Stata Blog: Not Elsewhere Classified.* https://blog.stata.com/2017/07/11/estimating-the-parameters-of-dsge-models/.

Also see

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

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

[DSGE] **Intro 3c** — Financial frictions model

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

[DSGE] **Intro 9a** — Bayesian estimation of a New Keynesian model

[DSGE] **dsge** — Linear dynamic stochastic general equilibrium models

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

Title

Intro 3b — New Classical model

Description Remarks and examples Reference Also see

Description

In this example, we solve a New Classical model similar to the one in King and Rebelo (1999). We also demonstrate how to compare a model's theoretical predictions under different parameter values using IRFs.

Remarks and examples

Remarks are presented under the following headings:

The model

In this model, output, consumption, investment, employment, and other variables are driven by state variables linked to production and demand. The model is similar to the one in King and Rebelo (1999) and is referred to as a real business cycle model.

The nonlinear form of the model is

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

$$H_t^{\eta} = \frac{W_t}{C_t} \tag{2}$$

$$Y_t = C_t + X_t + G_t \tag{3}$$

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

$$W_t = (1-\alpha)\frac{Y_t}{H_t} \tag{5}$$

$$R_t = \alpha\frac{Y_t}{K_t} \tag{6}$$

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

Equation (1) specifies consumption C_t as a function of expected future consumption and the expected future interest rate $E_t(R_{t+1})$. Equation (2) specifies labor hours H_t as a function of the wage W_t and consumption; it is a labor supply equation. Equation (3) is the national income accounting identity for a closed economy, specifying output Y_t as the sum of consumption, investment X_t, and government spending G_t. Equation (4) is a production function that specifies output as a function of labor input H_t, capital input K_t, and productivity Z_t. Equations (5) and (6) specify labor demand and capital demand, respectively. Equation (7) specifies the equation for capital accumulation. The model is completed when we add state transition equations for Z_t and G_t. These state transition equations are conventionally specified after the model has been linearized.

The linearized form of the model is

$$
\begin{aligned}
c_t &= E_t(c_{t+1}) - (1 - \beta + \beta\delta)E_t(r_{t+1}) \\
\eta h_t &= w_t - c_t \\
\phi_1 x_t &= y_t - \phi_2 c_t - g_t \\
y_t &= (1 - \alpha)(z_t + h_t) + \alpha k_t \\
w_t &= y_t - h_t \\
r_t &= y_t - k_t \\
k_{t+1} &= \delta x_t + (1 - \delta)k_t \\
z_{t+1} &= \rho_z z_t + \epsilon_{t+1} \\
g_{t+1} &= \rho_g g_t + \xi_{t+1}
\end{aligned}
$$

The model has six control variables and three state variables. Two of the state variables, z_{t+1} and g_{t+1}, are modeled as first-order autoregressive processes. The state equation for k_{t+1} depends on the current value of a control variable, namely, x_t.

Solving the model

The `solve` option of `dsge` places the model in state-space form without estimating parameters; it is similar to `iterate(0)` but is faster because it does not calculate standard errors. Using `solve` for different parameter values of your model is a useful way to explore the model's theoretical properties.

The parameter values used here are similar to those used in King and Rebelo (1999). Each has an interpretation. (1 − `alpha`) is labor's share of national income. `delta` is the depreciation rate of capital. `eta` is the slope of the labor supply curve. `phi1` and `phi2` are share parameters related to investment's share of national income and consumption's share of national income, respectively. `rhoz` and `rhog` are autoregressive parameters on the state variables.

```
. use https://www.stata-press.com/data/r18/usmacro2
(Federal Reserve Economic Data - St. Louis Fed, 2017-01-15)
. dsge (c = F.c - (1-{beta}+{beta}*{delta})*F.r, unobserved)
>      ({eta}*h = w - c, unobserved)
>      ({phi1}*x = y - {phi2}*c - g, unobserved)
>      (y = (1-{alpha})*(z+h) + {alpha}*k)
>      (w = y - h, unobserved)
>      (r = y - k, unobserved)
>      (F.k = {delta}*x+ (1-{delta})*k, state noshock)
>      (F.z = {rhoz}*z, state)
>      (F.g = {rhog}*g, state),
>      from(beta=0.96 eta=1 alpha=0.3 delta=0.025 phi1=0.2 phi2=0.6 rhoz=0.8 rhog=0.3)
>      solve noidencheck
DSGE model
Sample: 1955q1 thru 2015q4                          Number of obs = 244
Log likelihood = -1957.0261
```

y	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.96	.	.	.	.	.
delta	.025	.	.	.	.	.
eta	1	.	.	.	.	.
phi1	.2	.	.	.	.	.
phi2	.6	.	.	.	.	.
alpha	.3	.	.	.	.	.
rhoz	.8	.	.	.	.	.
rhog	.3	.	.	.	.	.
sd(e.z)	1	.			.	.
sd(e.g)	1	.			.	.

```
Note: Skipped identification check.
Note: Model solved at specified parameters.
```

The `solve` option solves the model at the specified values in `from()`. We skip the identification check with `noidencheck`. Simply solving the model does not involve any reference to the data or any estimation. Still, we can explore what these parameters imply.

Policy and transition matrices

After solving, we can use many of the postestimation commands, though standard errors will be missing throughout.

The state transition matrix shows how the state vector in the next period is related to the state vector in the current period. Some state variables are specified as first-order autoregressive processes, and their transition equations will simply repeat information that is already available in the estimation table. However, if any state variable equation contains control variables, then that state variable's transition equation will depend on the other state variables.

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

	Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
F.k						
k	.9256785	.	.	.	.	.
z	.1078282	.	.	.	.	.
g	-.1070547	.	.	.	.	.
F.z						
k	0	(omitted)				
z	.8	.	.	.	.	.
g	0	(omitted)				
F.g						
k	0	(omitted)				
z	0	(omitted)				
g	.3	.	.	.	.	.

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

The value of the state variables `z` and `g` in the next period depends only on their value in the current period, but the value of the capital stock `k` in the next period depends on the current value of all three state variables. This feature means that, for example, a shock to the `z` state variable has two effects: it increases future values of `z`, because `z` is autoregressive, but it also increases future values of `k`. Interrelationships among the state variables can generate more interesting patterns in the IRFs than the AR(1) dynamics that we saw in [DSGE] **Intro 3a**.

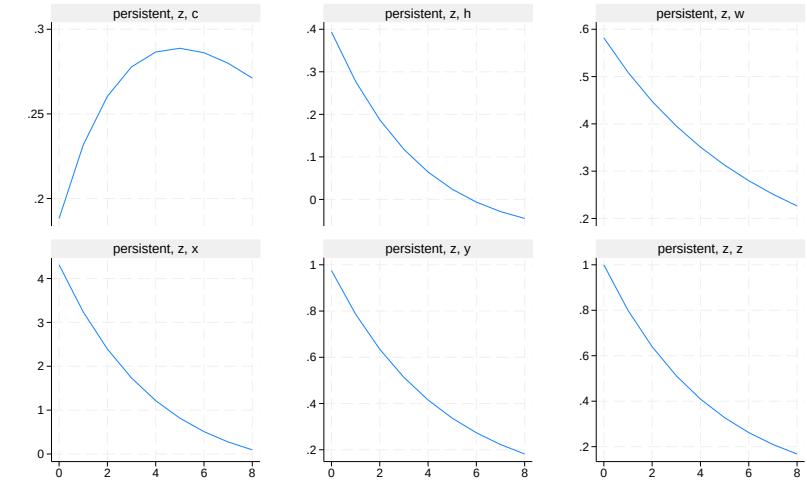
Impulse responses

One way to compare two parameter sets is to graph the impulse response of model variables to a shock under each parameter set. We first set the impulse–response file with `irf set` and then add impulse responses named `persistent` to the file with `irf create`.

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

. irf create persistent
(file rbcirf.irf updated)
```

The response of model variables to a shock to `z` is graphed by typing

```
. irf graph irf, irf(persistent) impulse(z) response(y c x h w z) noci
> byopts(yrescale)
```



Each graph is labeled with the IRF name, the impulse variable, and the response variable. For instance, the top-left graph shows the response of consumption to a shock to the z_t state variable. The bottom-right graph shows the response of the state variable z_t itself. The state is persistent, which is not surprising: we set the autoregressive parameter in the z_t equation to 0.8.

In the top-left graph, we see that consumption `c` rises over time before returning to steady state. The time unit is quarters, so a value of about 0.27, 4 periods after the shock, indicates that consumption is 0.27% above its steady-state value one year after the shock. Hours worked `h` are shown in the top center graph and rise initially before falling below steady state. The real wage `w`, output `y`, and investment `x` all rise.

Sensitivity analysis

The responses of variables to a shock to `z` are persistent. Some variables, like consumption and the wage, show dynamics beyond the simple autoregressive behavior of `z` itself. To evaluate the role of persistence in `z` on the persistence of other model variables, we rerun the `dsge` command. This time, we set the persistence of `z` to a smaller value of 0.6.

```
. dsge (c = F.c - (1-{beta}+{beta}*{delta})*F.r, unobserved)
>       ({eta}*h = w - c, unobserved)
>       ({phi1}*x = y - {phi2}*c - g, unobserved)
>       (y = (1-{alpha})*(z+h) + {alpha}*k)
>       (w = y - h, unobserved)
>       (r = y - k, unobserved)
>       (F.k = {delta}*x+ (1-{delta})*k, state noshock)
>       (F.z = {rhoz}*z, state)
>       (F.g = {rhog}*g, state),
>       from(beta=0.96 eta=1 alpha=0.3 delta=0.025 phi1=0.2 phi2=0.6 rhoz=0.6
>       rhog=0.3) solve noidencheck
DSGE model
Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -1659.7331
```

y	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.96	.	.	.	.	.
delta	.025	.	.	.	.	.
eta	1	.	.	.	.	.
phi1	.2	.	.	.	.	.
phi2	.6	.	.	.	.	.
alpha	.3	.	.	.	.	.
rhoz	.6	.	.	.	.	.
rhog	.3	.	.	.	.	.
sd(e.z)	1	.			.	.
sd(e.g)	1	.			.	.

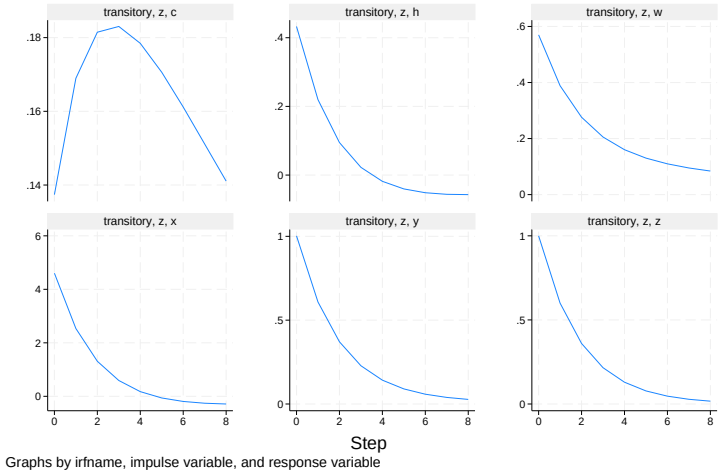
```
Note: Skipped identification check.
Note: Model solved at specified parameters.
```

The only change in the parameter set is that `rhoz` has been set to 0.6 from its earlier setting of 0.8. We can add the impulse responses of this model to the `irf` file with the name `transitory`,

```
. irf create transitory, replace
(irfname transitory not found in rbcirf.irf)
(file rbcirf.irf updated)
```

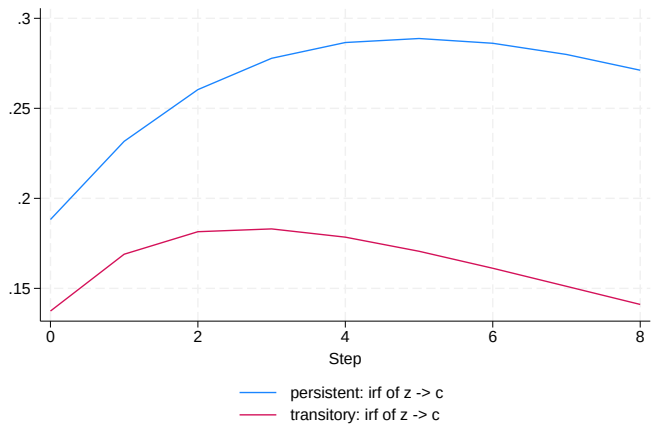
and graph them.

```
. irf graph irf, irf(transitory) impulse(z) response(y c x h w z) noci
> byopts(yrescale)
```



Model variables are much less persistent. We can use `irf ograph` to overlay the IRF for a variable under the two calibrations. This way we can view the differences across calibrations directly.

```
. irf ograph (persistent z c irf) (transitory z c irf)
```



When the shock itself is persistent, consumption responds persistently. When the shock is transitory, consumption returns to its steady-state value quickly.

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 3a** — New Keynesian model

[DSGE] **Intro 3c** — Financial frictions model

[DSGE] **Intro 3e** — Nonlinear New Classical model

[DSGE] **dsge** — Linear dynamic stochastic general equilibrium models

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

Title

Intro 3c — Financial frictions model

Description Remarks and examples Also see

Description

This introduction estimates and interprets the parameters of a model that incorporates financial frictions. The model is an extension of the one in [DSGE] **Intro 3a**.

Remarks and examples

Remarks are presented under the following headings:

The model

Equations (1)–(7) specify a model of financial frictions. The simplest such model places a wedge between two interest rates: the safe interest rate set by the central bank and the market interest rate that consumers and producers use.

$$\pi_t = \beta E_t \pi_{t+1} + \kappa x_t \tag{1}$$
$$x_t = E_t x_{t+1} - (i_t - E_t \pi_{t+1} - g_t) \tag{2}$$
$$r_t = \psi \pi_t + u_t \tag{3}$$
$$i_t = \chi r_t + e_t \tag{4}$$
$$g_{t+1} = \rho_g g_t + \xi_{t+1} \tag{5}$$
$$u_{t+1} = \rho_u u_t + \epsilon_{t+1} \tag{6}$$
$$e_{t+1} = \rho_e e_t + \eta_{t+1} \tag{7}$$

These are the linearized equations. The nonlinear equations are similar to those in *Writing down nonlinear DSGEs* in [DSGE] **Intro 1**.

This model is an extension of the one worked in [DSGE] **Intro 1**. It has an additional equation for the interest rate spread. As before, (1) specifies the inflation equation (a Phillips curve), (2) specifies the output gap equation (an Euler equation), and (3) specifies the equation for the safe interest rate (a Taylor rule). Equation (3) can be thought of as an equation that specifies the safe interest rate r_t. The new element is (4), which specifies an equation for the market interest rate i_t that enters the output gap equation. The market interest rate i_t is a function of the safe interest rate and a state variable e_t. The state variable e_t controls the interest rate spread and can be thought of as representing the state of the financial system. A large realization of e_t represents a large interest rate spread, indicating financial distress.

Parameter estimation

We estimate the parameters of the model in (1)–(7) using U.S. data on the Federal funds rate (safe rate set by the central bank), the high-grade corporate bond interest rate (a measure of market interest rates), and the inflation rate.

As is typical in these models, we constrain the parameter `beta` to 0.96.

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

. constraint 1 _b[beta]=0.96

. dsge (p = {beta}*F.p + {kappa}*x)
>      (x = F.x - (i - F.p - g), unobserved)
>      (i = {chi}*r + e)
>      (r = {psi}*p + u)
>      (F.e = {rhoe}*e, state)
>      (F.u = {rhou}*u, state)
>      (F.g = {rhoz}*g, state),
>      from(psi=2 chi=0.8) constraint(1)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -4780.2037
Iteration 1:  Log likelihood = -1731.5956  (backed up)
Iteration 2:  Log likelihood = -1315.8819  (backed up)
Iteration 3:  Log likelihood = -1161.0796  (backed up)
Iteration 4:  Log likelihood = -1115.2257  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -1069.1254  (backed up)
Iteration 6:  Log likelihood = -1008.6739  (not concave)
Iteration 7:  Log likelihood = -933.21643  (not concave)
Iteration 8:  Log likelihood = -924.30496  (not concave)
Iteration 9:  Log likelihood =  -916.3707  (not concave)
Iteration 10: Log likelihood = -911.96573  (not concave)
Iteration 11: Log likelihood = -897.31045
Iteration 12: Log likelihood = -889.39375
Iteration 13: Log likelihood = -883.98729
Iteration 14: Log likelihood = -882.41002
Iteration 15: Log likelihood = -882.14866
Iteration 16: Log likelihood = -882.13398
Iteration 17: Log likelihood = -882.13195
Iteration 18: Log likelihood = -882.13195
```

```
DSGE model
Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -882.13195
 ( 1)  [/structural]beta = .96
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.96	(constrained)				
kappa	.0503257	.0184939	2.72	0.007	.0140783	.086573
chi	.9067394	.1249768	7.26	0.000	.6617894	1.151689
psi	6.332377	2.567617	2.47	0.014	1.29994	11.36481
rhoe	.8478222	.0301221	28.15	0.000	.788784	.9068605
rhou	.815346	.033254	24.52	0.000	.7501694	.8805226
rhoz	.9861866	.0099767	98.85	0.000	.9666327	1.00574
sd(e.e)	.8857071	.1343023			.6224795	1.148935
sd(e.u)	7.160717	2.933797			1.41058	12.91085
sd(e.g)	.3911862	.0358352			.3209506	.4614219

The persistence of the financial shock `rhoe` is estimated to be 0.85. The slope of the Phillips curve, `kappa`, is somewhat flatter in this model than in the one in [DSGE] **Intro 1**. The coefficient on inflation in the interest rate equation is 6.3 and indicates that the central bank increases interest rates much more than one for one in response to movements in inflation.

Policy and transition matrices

We can read off the impact effect of shocks using the policy matrix. The response to the financial shock `e` will be of most interest.

```
. estat policy
Policy matrix
```

		Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
p							
	e	-.1832608	.0740949	-2.47	0.013	-.3284842	-.0380374
	u	-.1584194	.0641353	-2.47	0.014	-.2841223	-.0327165
	g	.2096327	.1003828	2.09	0.037	.0128861	.4063794
x							
	e	-.6776486	.3498876	-1.94	0.053	-1.363416	.0081185
	u	-.683934	.2783442	-2.46	0.014	-1.229479	-.1383894
	g	.2218595	.1284728	1.73	0.084	-.0299427	.4736616
i							
	e	-.0522495	.0403212	-1.30	0.195	-.1312776	.0267785
	u	-.0028755	.0041244	-0.70	0.486	-.0109591	.0052082
	g	1.203672	.0979344	12.29	0.000	1.011724	1.39562
r							
	e	-1.160476	.1361174	-8.53	0.000	-1.427261	-.8936912
	u	-.0031712	.0045309	-0.70	0.484	-.0120517	.0057092
	g	1.327473	.222489	5.97	0.000	.8914029	1.763544

Importantly, the financial shock causes `r` to fall by more than `i`, indicating an increase in the interest rate spread. Inflation also falls, as does the output gap. It is the fall in the inflation rate that causes the central bank's interest rate to fall on impact of the shock.

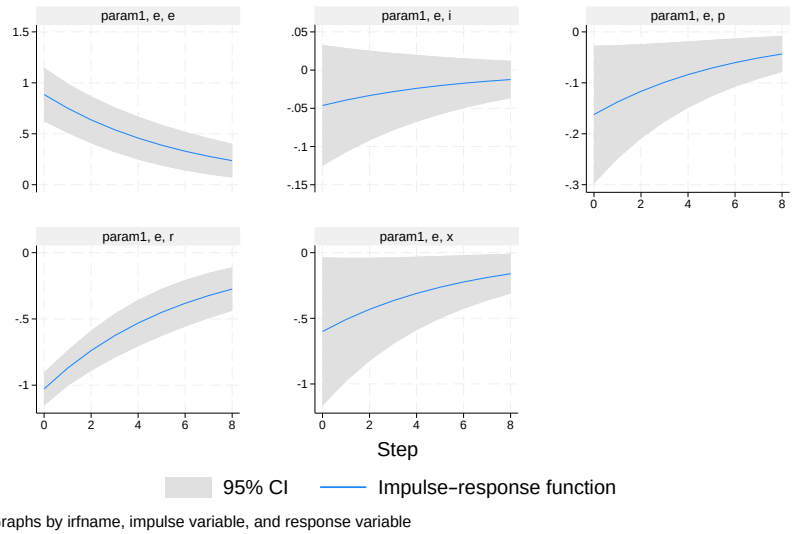
Impulse responses

We use the `irf set` command to set `finirf.irf` as the active IRF file and then use `irf create` to create and store the impulse responses under the name `param`.

```
. irf set finirf
(file finirf.irf created)
(file finirf.irf now active)

. irf create param1
(file finirf.irf updated)
```

Finally, we graph the impulse response to a financial shock.

```
. irf graph irf, irf(param1) impulse(e) response(e x p i r) byopts(yrescale)
```



Graphs by irfname, impulse variable, and response variable

The top middle panel shows the effect of the shock to the `e` equation on `e` itself. The shock represents a persistent increase in the spread between the two interest rates. The increased interest rate spread causes both the output gap and inflation rate to fall. The safe interest rate `r` falls dramatically, and the combined effect of the increased spread `e` and reduced safe interest rate `r` causes the market interest rate `i` to be little changed.

Also see

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

[DSGE] **Intro 3a** — New Keynesian model

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

[DSGE] **dsge** — Linear dynamic stochastic general equilibrium models

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

Title

Intro 3d — Nonlinear New Keynesian model

Description Remarks and examples Reference Also see

Description

This introduction estimates and interprets the parameters of a nonlinear New Keynesian model. We demonstrate the effect of a change in constraints on a model that is only partially identified.

Remarks and examples

Remarks are presented under the following headings:

The model

Equations (1)–(5) specify a simplified nonlinear New Keynesian model of inflation Π_t, the output gap X_t, and the interest rate R_t. These three variables are driven by two state variables, Z_t and M_t. Recall that in this manual, capital letters denote the level of a variable, while lowercase letters denote deviations from steady state.

$$1 = \beta E_t \left(\frac{X_t}{X_{t+1}} \frac{1}{Z_t} \frac{R_t}{\Pi_{t+1}} \right) \tag{1}$$

$$(\theta - 1) + \phi(\pi_t - 1)\pi_t = \theta X_t + \phi\beta E_t \left\{ (\pi_{t+1} - 1)\pi_{t+1} \right\} \tag{2}$$

$$\beta R_t = \Pi_t^{\psi} M_t \tag{3}$$

$$\ln(M_{t+1}) = \rho_m \ln(M_t) + u_{t+1} \tag{4}$$

$$\ln(Z_{t+1}) = \rho_z \ln(Z_t) + e_{t+1} \tag{5}$$

Equation (1) is the Euler equation linking the output gap in the current period to the expected future output gap and the real interest rate. In this equation, Z_t can be interpreted as a state variable influencing household decisions. In applications, Z_t stands in for the natural rate of interest, consumer spending shocks, or government expenditures. Equation (2) is the Phillips curve linking the current rate of inflation to expected future inflation and the output gap. Equation (3) is the interest-rate rule linking the interest rate to the inflation rate; in this equation, the state variable M_t captures all movements in the interest rate that are due to factors other than inflation. Equation (4) specifies the stochastic process for the state variable in the interest rate equation. Equation (5) specifies the stochastic process for the state variable in the Euler equation.

Parameter estimation

We estimate the model's parameters using data on the inflation rate and the nominal interest rate.

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

Not all model parameters are identified. We constrain β to be 0.96, a common value in the literature. The parameters θ and ϕ are collinear and cannot be estimated together. The parameter θ is set to 5, another common value in the literature. Later, we will explore consequences of the choice $\theta = 5$. The remaining six parameters are identified.

```
. constraint 1 _b[theta]=5
. constraint 2 _b[beta]=0.96
```

To specify this model to `dsgenl`, we will write each equation and then specify four options. The first three options tell `dsgenl` about the model's structure. The `observed()` option specifies observed control variables that are to be used in estimation. The `unobserved()` option specifies latent control variables. The `exostate()` option specifies the collection of state variables. The other option we use is the standard `constraints()` option that applies our constraints on the parameters `beta` and `theta`.

```
. dsgenl (1 = {beta}*(x/F.x)*(1/z)*(r/F.p))
>     ({theta}-1 + {phi}*(p -1)*p =  {theta}*x + {phi}*{beta}*(F.p-1)*F.p)
>     (({beta})*r = (p)^({psi=2})*m)
>     (ln(F.m) = {rhom}*ln(m))
>     (ln(F.z) = {rhoz}*ln(z)),
>     exostate(z m) unobserved(x) observed(p r)
>     constraint(1 2) nolog
Solving at initial parameter vector ...
Checking identification ...

First-order DSGE model

Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -753.57131
 ( 1)  [/structural]theta = 5
 ( 2)  [/structural]beta = .96
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.96	(constrained)				
theta	5	(constrained)				
phi	47.07939	15.9416	2.95	0.003	15.83443	78.32435
psi	1.943008	.2957895	6.57	0.000	1.363271	2.522745
rhom	.7005489	.0452605	15.48	0.000	.61184	.7892577
rhoz	.9545255	.0186424	51.20	0.000	.9179871	.991064
sd(e.z)	.5689908	.0982979			.3763305	.761651
sd(e.m)	2.318208	.3047461			1.720916	2.915499

Because this is the nonlinear version of the model in [DSGE] **Intro 3a**, it is instructive to compare the two results. The two models deliver nearly identical estimates of most parameters; the small differences seen in the fourth and higher decimal places are due to differences in the numerical techniques used in each command. The only parameter that differs for the two models is the stickiness parameter `phi`, which is the counterpart of the Phillips curve slope `kappa` from [DSGE] **Intro 3a**. The parameterization of the Phillips curve in (2) above differs from the one in [DSGE] **Intro 3a**. However, the two are linked by the relationship $\kappa = (\theta - 1)/\phi$, and we can use that relationship to show that the models are equivalent.

```
. nlcom (_b[theta] - 1)/_b[phi]
       _nl_1: (_b[theta] - 1)/_b[phi]
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
_nl_1	.0849629	.0287694	2.95	0.003	.028576	.1413498

The value above is the same as the coefficient for `kappa` in [DSGE] **Intro 3a**.

Policy and transition matrices

The policy matrix is part of the state-space representation of a DSGE model. It expresses the control variables of the model as functions of the state variables alone. Elements of the policy matrix represent the response of a control variable to a one-unit increase in a state variable. In a linear DSGE model, the policy matrix will also be linear. For a nonlinear DSGE model, the policy function will typically be nonlinear. The policy matrix reported after `dsgenl` is a linear approximation of the true, nonlinear policy matrix evaluated at the estimated parameter vector. For details on the linearization of the policy function, see DeJong and Dave (2011, sec. 5.3).

```
. estat policy
Policy matrix
```

		Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
x							
	z	.952921	.4813967	1.98	0.048	.0094009	1.896441
	m	-1.608216	.405057	-3.97	0.000	-2.402113	-.814319
p							
	z	.9678135	.277745	3.48	0.000	.4234433	1.512184
	m	-.4172515	.0393611	-10.60	0.000	-.494398	-.3401051
r							
	z	1.880469	.2615995	7.19	0.000	1.367744	2.393195
	m	.189277	.0591663	3.20	0.001	.0733133	.3052407

An increase in `m` decreases inflation, decreases the output gap, and increases interest rates. We may interpret it as an unexpected contractionary monetary policy. An increase in `z` increases inflation, increases the output gap, and increases interest rates.

Next, we can look at the state transition matrix. Because the states are uncorrelated with each other in this example, the elements of the state transition matrix are the persistence parameters in the model.

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

		Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
F.z							
	z	.9545255	.0186424	51.20	0.000	.9179871	.991064
	m	0	(omitted)				
F.m							
	z	0	(omitted)				
	m	.7005489	.0452605	15.48	0.000	.61184	.7892577

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

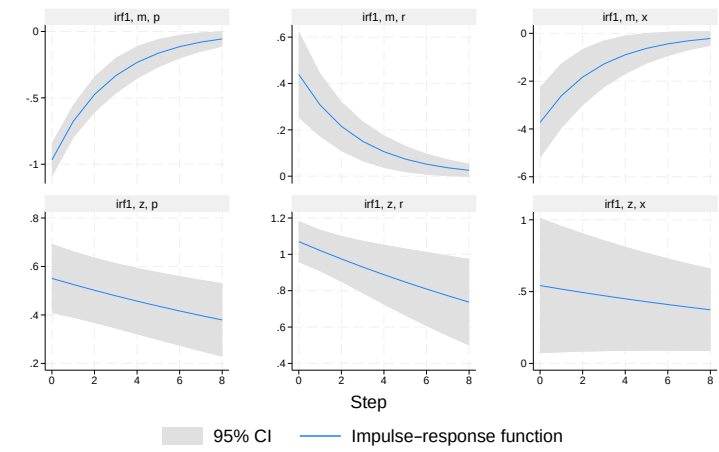
These parameters repeat the results we saw in the estimation table. When the state variables are correlated, the transition matrix will reveal new information about the correlations among state variables.

Impulse responses

The impulse–response function traces out the effect of a shock on the model's state and control variables. We use two commands to set up the impulse responses: an `irf set` command to set the active IRF file and an `irf create` command to create the impulse–response functions.

From there, we can use `irf graph irf` to graph the impulse–response functions.

```
. irf set nlex1irf, replace
(file nlex1irf.irf created)
(file nlex1irf.irf now active)
. irf create irf1, replace
(irfname irf1 not found in nlex1irf.irf)
(file nlex1irf.irf updated)
. irf graph irf, impulse(m z) response(x p r) byopts(yrescale)
```



Graphs by irfname, impulse variable, and response variable

Responses to a monetary shock are displayed in the first row. A shock to monetary policy leads inflation to fall, the interest rate to rise, and the output gap to fall. From the bottom row, we see that an increase in the natural rate of interest increases all three control variables.

A change in constraints

In the above example, the parameter θ was set to 5 to achieve identification of the other parameters. The following example shows the consequences of this decision by examining a case where θ was fixed to a different value. Suppose that instead of setting $\theta = 5$ above, we set $\theta = 2$. How would the change in constraints change our estimates? First, we store the current estimates with `estimates store`. We call the current estimates `theta5` because they represent the case where $\theta = 5$.

```
. dsgenl
First-order DSGE model
Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -753.57131
 ( 1)  [/structural]theta = 5
 ( 2)  [/structural]beta = .96
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.96	(constrained)				
theta	5	(constrained)				
phi	47.07939	15.9416	2.95	0.003	15.83443	78.32435
psi	1.943008	.2957895	6.57	0.000	1.363271	2.522745
rhom	.7005489	.0452605	15.48	0.000	.61184	.7892577
rhoz	.9545255	.0186424	51.20	0.000	.9179871	.991064
sd(e.z)	.5689908	.0982979			.3763305	.761651
sd(e.m)	2.318208	.3047461			1.720916	2.915499

```
. estimates store theta5
```

We can now recall these estimates later as needed. We set up the new constraint with

```
. constraint 3 _b[theta]=2
```

and reestimate parameters. The `nolog` option is used to suppress the iteration log.

```
. dsgenl (1 = {beta}*(x/F.x)*(1/z)*(r/F.p))
>      ({theta}-1 + {phi}*(p -1)*p =  {theta}*x + {phi}*{beta}*(F.p-1)*F.p)
>      (({beta})*r = (p)^({psi=2})*m)
>      (ln(F.m) = {rhom}*ln(m))
>      (ln(F.z) = {rhoz}*ln(z)),
>      exostate(z m) unobserved(x) observed(p r)
>      constraint(2 3) nolog
Solving at initial parameter vector ...
Checking identification ...

First-order DSGE model

Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -753.57131
 ( 1)  [/structural]beta = .96
 ( 2)  [/structural]theta = 2
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.96	(constrained)				
theta	2	(constrained)				
phi	11.76979	3.985338	2.95	0.003	3.958675	19.58091
psi	1.943004	.2957869	6.57	0.000	1.363272	2.522736
rhom	.7005482	.0452601	15.48	0.000	.6118401	.7892563
rhoz	.9545256	.0186424	51.20	0.000	.9179872	.991064
sd(e.z)	.5689892	.0982974			.3763299	.7616486
sd(e.m)	2.318204	.3047434			1.720918	2.91549

```
. estimates store theta2
```

We saved these estimates in `theta2` because they represent the case where $\theta = 2$. `estimates table` displays the two results. The `stats(ll)` option displays the estimated log likelihood with the estimated structural parameters.

```
. estimates table theta5 theta2, b(%9.4f) stats(ll)
```

Variable	theta5	theta2
/structural		
beta	0.9600	0.9600
theta	5.0000	2.0000
phi	47.0794	11.7698
psi	1.9430	1.9430
rhom	0.7005	0.7005
rhoz	0.9545	0.9545
sd(e.z)	0.5690	0.5690
sd(e.m)	2.3182	2.3182
Statistics		
ll	-753.5713	-753.5713

The reported log-likelihood values are identical across the two parameterizations. The change in constraints slides us along the top of a ridge in the likelihood function. The estimated value of `phi` differs when the constraint is changed, but all other parameter estimates are identical. Even though the point estimate of `phi` has changed, the z statistic and p-value are unchanged.

As discussed above, the underlying parameter that is being estimated here is $\kappa = (\theta - 1)/\phi$. The restrictions on the parameter θ lead to a change in the estimated value of ϕ but do not change the

estimated value of κ. We can use `nlcom` to recover κ first from the model with $\theta = 5$ and then from the model with $\theta = 2$.

```
. estimates restore theta5
(results theta5 are active now)
. nlcom (_b[theta]-1) / _b[phi]
       _nl_1: (_b[theta]-1) / _b[phi]
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
_nl_1	.0849629	.0287694	2.95	0.003	.028576	.1413498

```
. estimates restore theta2
(results theta2 are active now)
. nlcom (_b[theta]-1) / _b[phi]
       _nl_1: (_b[theta]-1) / _b[phi]
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
_nl_1	.0849633	.0287692	2.95	0.003	.0285767	.1413498

The equivalence of the two parameterizations can also be seen through the policy matrix. The estimated policy matrix when $\theta = 2$ is

```
. estat policy
Policy matrix
```

	Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
x						
z	.9529194	.4813921	1.98	0.048	.0094082	1.896431
m	-1.608214	.4050521	-3.97	0.000	-2.402102	-.8143265
p						
z	.9678173	.2777448	3.48	0.000	.4234475	1.512187
m	-.4172521	.0393609	-10.60	0.000	-.4943981	-.3401061
r						
z	1.880473	.2615995	7.19	0.000	1.367747	2.393199
m	.1892774	.059166	3.20	0.001	.0733142	.3052406

This policy matrix is identical to the one shown above, in which the constraint $\theta = 5$ was applied. The two parameter vectors imply the same reduced-form matrices.

Reference

DeJong, D. N., and C. Dave. 2011. *Structural Macroeconometrics*. 2nd ed. Princeton, NJ: Princeton University Press.

Also see

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

[DSGE] **Intro 3a** — New Keynesian model

[DSGE] **Intro 3e** — Nonlinear New Classical model

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

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

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

Title

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

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.
```

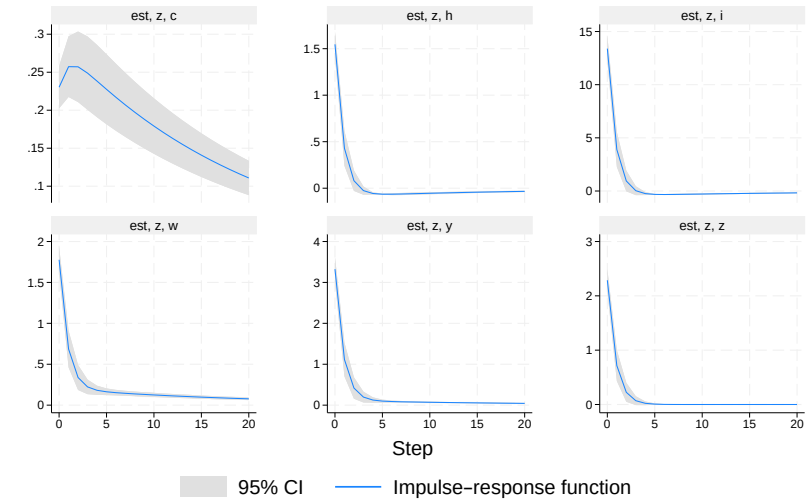
The "own" effects (`k` on `F.k` and `z` on `F.z`) are persistence parameters. The other reported effects show how a change in one state variable contemporaneously affects another. A change in the capital stock does not affect future productivity, but a change in productivity does affect the future capital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

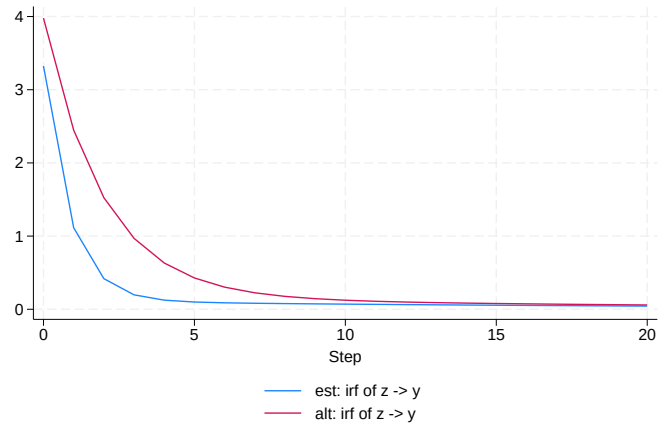
We passed the parameter vector in using `from()`, and we used the `solve` option to prevent estimation. Because we specified the `solve` option, the estimation table simply repeats the values we fed in with `from()`. Of course, because we did not estimate, we do not get standard errors. When we use `solve`, 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.

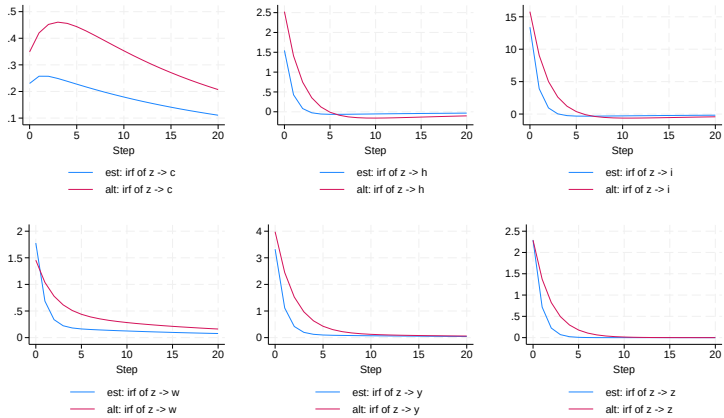
```
. irf ograph (est z y irf) (alt z y irf)
```



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

Title

Intro 3f — Stochastic growth model

Description Remarks and examples References Also see

Description

In this introduction, we solve a stochastic growth model and interpret objects that are available after solving. The topics discussed in this introduction are more technical than those in the introductions describing other classic models. We provide some details about how `dsgenl` finds the model's steady state, we describe how it solves for a first-order approximation to the model's state-space form, and we provide guidance on the differences between and interpretations of linear versus log-linear approximations. For Bayesian analysis of this model, see [DSGE] **Intro 9b**.

Remarks and examples

Remarks are presented under the following headings:

The model

The model contains equations that jointly determine output Y_t, the interest rate R_t, consumption C_t, capital K_t, and productivity Z_t. The model contains four parameters: α, β, δ, and ρ. This model is a variant on the model used in Schmitt-Grohé and Uribe (2004).

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

$$Y_t = Z_t K_t^{\alpha} \tag{2}$$

$$R_t = \alpha Z_t K_t^{\alpha-1} \tag{3}$$

$$K_{t+1} = Y_t - C_t + (1-\delta)K_t \tag{4}$$

$$\ln(Z_{t+1}) = \rho \ln(Z_t) + e_{t+1} \tag{5}$$

Equation (1) defines a relationship between consumption growth C_{t+1}/C_t and the real interest rate R_{t+1}. Equation (2) is a production function for output Y_t as a function of productivity Z_t and capital K_t. Equation (3) is a model for the interest rate. Equation (4) is the equation for capital accumulation; capital in the next period is equal to undepreciated capital this period $(1-\delta)K_t$ plus unconsumed output $Y_t - C_t$. Equation (5) is a law of motion for productivity Z_t. The parameter β is a discount factor in the consumption equation, the parameter α is a production parameter in the output equation, the parameter δ is a depreciation parameter in the capital equation, and the parameter ρ is a persistence parameter in the productivity equation.

The state variables are the current-period capital stock and the level of productivity, (K_t, Z_t). The control variables are consumption, the interest rate, and output (C_t, R_t, Y_t). To solve the model means to write the control variables as functions of the state variables and to write the future values of the state variables as functions of the current state variables. Equation (2) is already solved, for example, because it writes output as a function of the state variables alone. But (4) is not a reduced form, because it writes the future capital stock as a function of the state variable K_t and the control variables C_t and Y_t.

The model equations have two key features. First, the model is nonlinear in its variables. Second, the model contains expectations of variables at time $t+1$, conditional on information at time t. `dsgenl` solves the model by taking a linear (or log-linear) approximation to the model equations at the steady state. Two concepts need to be clarified: the steady state and the approximation. We will describe these two issues in general, then return to the specific model described above.

Approximating the solution to a nonlinear DSGE model

We can gather up all the model equations into a vector of functions, $\mathbf{f}$. We denote the model's vector of state variables by $\mathbf{x}_t$ and the vector of control variables by $\mathbf{y}_t$. Now, the DSGE model can be written as

$$E_t\left\{\mathbf{f}(\mathbf{x}_{t+1}, \mathbf{y}_{t+1}, \mathbf{x}_t, \mathbf{y}_t, \boldsymbol{\theta})\right\} = \mathbf{0}$$

The expectation operator applies to the entire system of equations. In a generic DSGE model, there are n equations, n_x state variables, and n_y control variables; it must be the case that $n_x + n_y = n$.

The steady state of the model is the vector of values to which model variables converge in the absence of shocks. This is equivalent to the solution to the model's system of equations after setting all future values of variables equal to current values and dropping the expectation operator. This location is a vector of numbers $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ that solves

$$\mathbf{f}(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{x}}, \overline{\mathbf{y}}, \boldsymbol{\theta}) = \mathbf{0}$$

This is a system of n equations in n unknowns, and its solution is the steady state. It is this point around which we take our approximation. The location of the steady state is influenced by the model parameters $\boldsymbol{\theta}$. It is possible that the parameter vector you specify does not admit a steady state; in such a case, `dsgenl` will halt and issue an error message. You may specify starting values for the search for the steady state with the `steadyinit()` option.

We take an approximation to the model equations at the steady state. Differentiate $\mathbf{f}$ with respect to each variable at the point $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{x}}, \overline{\mathbf{y}})$. Differentiation generates a linearized model that is described by four matrices, namely,

$$\mathbf{F}_1 E_t\widehat{\mathbf{x}}_{t+1} + \mathbf{F}_2 E_t\widehat{\mathbf{y}}_{t+1} + \mathbf{F}_3\widehat{\mathbf{x}}_t + \mathbf{F}_4\widehat{\mathbf{y}}_t = \mathbf{0}$$

The matrix $\mathbf{F}_1$ contains partial derivatives of each model equation with respect to one-period-ahead state variables. The other $\mathbf{F}_i$ matrices are defined similarly. Each element of each $\mathbf{F}_i$ matrix depends on the structural parameter vector $\boldsymbol{\theta}$. There are two ways of obtaining the $\mathbf{F}_i$ matrices: linear approximation and log-linear, or percentage, approximation. The "hat" notation denotes the variables of the linear model. In a linear approximation,

$$\widehat{\mathbf{x}}_t = (\mathbf{x}_t - \overline{\mathbf{x}})$$

that is, $\widehat{\mathbf{x}}_t$ is the unit difference from steady state. In a log-linear approximation,

$$\widehat{\mathbf{x}}_t = (\mathbf{x}_t - \overline{\mathbf{x}})/\overline{\mathbf{x}}$$

so that $\widehat{\mathbf{x}}_t$ is measured in percentage difference from the steady state. Which approximation is applied will affect the $\mathbf{F}_i$ matrices, parameter estimation, and the interpretation of all objects available after estimation.

Hatted variables are measured as deviations from steady state. At this point, the model is in the form of a linear DSGE model, and the methods of Klein (2000) may be employed to find the solution. The stability conditions of Klein (2000) apply to the $\mathbf{F}_i$ matrices, and `dsgenl` will not proceed if the stability conditions are not met. If the stability conditions are met, we exactly solve the linear model. The solution is of the form

$$\widehat{\mathbf{y}}_t = \mathbf{G}\widehat{\mathbf{x}}_t$$
$$\widehat{\mathbf{x}}_{t+1} = \mathbf{H}\widehat{\mathbf{x}}_t + \mathbf{M}\mathbf{e}_{t+1}$$

Because shocks $\mathbf{e}_t$ are mean zero, it does not matter whether we write $\mathbf{e}_t$ or $\widehat{\mathbf{e}}_t$.

In a nonlinear DSGE model, the linear solution matrices are a linear approximation to the true nonlinear solution functions,

$$\mathbf{y}_t = \mathbf{g}(\mathbf{x}_t)$$
$$\mathbf{x}_{t+1} = \mathbf{h}(\mathbf{x}_t) + \mathbf{M}\mathbf{e}_{t+1}$$

The matrix $\mathbf{G}$ is an approximation to the function $\mathbf{g}(\cdot)$, and the matrix $\mathbf{H}$ is an approximation to the function $\mathbf{h}(\cdot)$. Once the solution has been obtained, we can view the steady state, the approximate solution matrices, the impulse responses, and other objects.

Specifying the model to Stata

We import U.S. macro data. Although we are solving the model (hence, the data will play no role), `dsgenl` expects there to be data anyway.

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

Next, we set up a parameter matrix containing the values that we will use for our model. We type

```
. matrix param_mat = (0.33, 0.96, 0.025, 0.9, 1)
. matrix colnames param_mat = alpha beta delta rho /sd(e.z)
```

These values are similar to those in King and Rebelo (1999) and Schmitt-Grohé and Uribe (2004). The elasticity of output with respect to capital α is set to 0.33. The discount factor β is set to 0.96. The depreciation rate δ is set to 0.025. The persistence of productivity is set to 0.9, and the standard deviation of the shock to productivity is set to 1.

Next, we specify the model.

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

First-order DSGE model

Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -2114.8044
```

y	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.96	.	.	.	.	.
delta	.025	.	.	.	.	.
alpha	.33	.	.	.	.	.
rho	.9	.	.	.	.	.
sd(e.z)	1	.			.	.

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

Each equation is bound in parentheses. Variables appear without any special modification; parameters are bound in parentheses. After the equation list, we specify seven options. The first four tell `dsgenl` how to interpret each variable in the model. The control variables `y`, `r`, and `c` are specified as being either `observed()` or `unobserved()`. Because we are only solving the model, it is inconsequential how we place control variables into these categories. In an estimation exercise, data on observed control variables are used to estimate model parameters, and which variables are observable can affect parameter identification. The state variables are divided into categories according to whether they are subject to shocks. The state variable `k` is not subject to shocks; it is an endogenous state variable. The state variable `z` is subject to shocks; it is an exogenous state variable.

The `from()` option specifies the starting values for the parameters. The `solve` option declares that we wish to solve the model at the values in `from()`. The `noidencheck` option skips the check for local identification.

The output table repeats the parameter vector that we passed in; it provides no new information. The notes below the estimation table remind us that we have skipped the identification check and that we have solved the model rather than estimated the model parameters.

After solving

After solving the model by running `dsgenl`, you can view the steady state, the approximate policy and transition matrices, and more using the `estat` commands.

The steady state

In the process of finding the solution, `dsgenl` finds the steady state of the model. The steady state is displayed with

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

	Coefficient
k	10.88
z	1.00
c	1.93
r	0.07
y	2.20

Because we used `dsgenl`'s `solve` option, `estat steady` displays the steady state of the model under the initial parameter vector. When `solve` is not specified, `estat steady` displays the steady state of the model at the converged parameter values.

The steady-state level of the capital stock is 10.88 units, and the steady-state level of output is 2.20 units. The steady-state interest rate is 0.07, or 7%. Some of these values are more meaningful when expressed as ratios rather than when they are expressed as raw values. The model's consumption-to-output ratio is about 0.88. The model's capital-to-output ratio is about 4.9.

Approximations to the policy and transition matrices

Next, we approximate the policy and state transition matrices. `estat policy` provides the approximate policy matrix.

```
. estat policy, compact
Policy matrix
```

	k	z
c	.6077069	.3431366
r	-.67	1
y	.33	1

By default, `dsgenl` uses a log-linear approximation, and the results of `estat policy` are also based on this approximation. This means that variables are measured as percentage deviations from steady state. The interpretation of the output equation in the policy matrix is that

$$\frac{Y_t - \overline{Y}}{\overline{Y}} \approx 0.33\frac{K_t - \overline{K}}{\overline{K}} + \frac{Z_t - \overline{Z}}{\overline{Z}}$$

A log-linear approximation was chosen because many DSGE model equations are linear in logs, or nearly so. For instance, in logs, (2) is

$$\log(Y_t) = \alpha\log(K_t) + \log(Z_t)$$

just as the policy matrix describes.

The entries in the policy matrix are impact effects. They display the percentage change in a control variable that results from a one-percent change in a state variable. From the consumption equation, a 1% increase in z causes consumption to rise by 34%. From the output equation, a 1% increase in z causes output to rise by 1%.

estat transition provides the approximate state transition matrix.

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

	k	z
k		
F1.	.9340903	.1412781
z		
F1.	0	.9

The state transition matrix displays how state variables in the next period depend on state variables today. A 1% increase in z causes capital to rise by 0.14% one period in the future. Notice that an increase in z has an effect on k, but an increase in k has no cross-effect on z.

Linear and log-linear approximations

Specifying option linearapprox causes dsgenl to take a linear approximation rather than a log-linear approximation. We first re-solve the model.

```
. dsgenl (1 = {beta}*(F.c/c)^(-1)*(1+F.r-{delta}))
>        (r = {alpha}*z*k^({alpha}-1))
>        (y = z*k^{alpha})
>        (F.k = y - c + (1-{delta})*k)
>        (ln(F.z) = {rho}*ln(z))
>        , observed(y) unobserved(c r) exostate(z) endostate(k)
>          from(param_mat) solve noidencheck linearapprox
Solving at initial parameter vector ...
First-order DSGE model
Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -808.47167
```

y	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.96	.	.	.	.	.
delta	.025	.	.	.	.	.
alpha	.33	.	.	.	.	.
rho	.9	.	.	.	.	.
sd(e.z)	1	.			.	.

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

We next examine the impact that the `linearapprox` option has on postsolution objects. The choice of linear or log-linear approximation does not affect the location of the steady state.

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

	Coefficient
k	10.88
z	1.00
c	1.93
r	0.07
y	2.20

This is to be expected; the steady state is the point around which approximations are taken, but its location does not depend on the choice of approximation.

The policy matrix has changed.

```
. estat policy, compact
Policy matrix
```

	k	z
c	.1075764	.6610193
r	-.0041045	.0666667
y	.0666667	2.198462

Each entry displays the unit increase in a control variable after a unit increase in a state variable. As an example, we use the estimates in the last line of this table. The interpretation of the output equation is

$$(Y_t - \overline{Y}) \approx 0.067(K_t - \overline{K}) + 2.2(Z_t - \overline{Z})$$

so that a one-unit increase in the capital stock leads output to rise by 0.067 units. The other entries are interpreted analogously.

The state transition matrix is now

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

	k	z
k		
F1.	.9340903	1.537443
z		
F1.	0	.9

The approximate capital accumulation equation is

$$(K_{t+1} - \overline{K}) \approx 0.93(K_t - \overline{K}) + 1.54(Z_t - \overline{Z})$$

so that a one-unit increase in productivity Z_t leads to a 1.54-unit increase in the capital stock in the next period.

References

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.

Klein, P. 2000. Using the generalized Schur form to solve a multivariate linear rational expectations model. *Journal of Economic Dynamics and Control* 24: 1405–1423. https://doi.org/10.1016/S0165-1889(99)00045-7.

Schmitt-Grohé, S., and M. Uribe. 2004. Solving dynamic general equilibrium models using a second-order approximation to the policy function. *Journal of Economic Dynamics and Control* 28: 755–775. https://doi.org/10.1016/S0165-1889(03)00043-5.

Also see

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

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

[DSGE] **Intro 3e** — Nonlinear New Classical model

[DSGE] **Intro 9b** — Bayesian estimation of stochastic growth model

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

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

Title

Intro 4 — Writing a DSGE in a solvable form

Description Remarks and examples Also see

Description

Many DSGE models, when written in their natural representation based on economic theory, include problematic terms that do not fit into the algebraic form required to solve a DSGE model. This entry shows how to accommodate these problematic terms by defining a new state variable or a new control variable and then rewriting the equations so that they have the required form. We demonstrate how to make the required changes using `dsge`, but the advice in this entry applies to both `dsge` and `dsgenl`.

We assume that you are already familiar with the elements of a DSGE; see [DSGE] **Intro 1**.

Remarks and examples

Remarks are presented under the following headings:

Introduction

We can estimate the parameters of a DSGE model only if we can solve for its state-space form, and we can solve a DSGE model for its state-space form only if the equations for control variables and state variables have specific structures. Unfortunately, many DSGE models contain problematic terms that do not fit into these structures without some manipulation. Fortunately, the manipulation is easy, once you know the tricks discussed below.

As we discussed in *Structural and reduced forms of DSGE models* in [DSGE] **Intro 1**, the structural form of a DSGE model can be written as

$$E_t[\mathbf{f}(\mathbf{x}_{t+1}, \mathbf{y}_{t+1}, \mathbf{x}_t, \mathbf{y}_t, \boldsymbol{\theta})] = \mathbf{0} \tag{1}$$

and the structural form of a linearized DSGE model can be written as

$$\mathbf{A}_0\mathbf{y}_t = \mathbf{A}_1 E_t(\mathbf{y}_{t+1}) + \mathbf{A}_2\mathbf{y}_t + \mathbf{A}_3\mathbf{x}_t \tag{2}$$
$$\mathbf{B}_0\mathbf{x}_{t+1} = \mathbf{B}_1 E_t(\mathbf{y}_{t+1}) + \mathbf{B}_2\mathbf{y}_t + \mathbf{B}_3\mathbf{x}_t + \mathbf{C}\boldsymbol{\epsilon}_{t+1} \tag{3}$$

where $\mathbf{y}_t$ is a vector of control variables, $\mathbf{x}_t$ is a vector of state variables, and $\boldsymbol{\epsilon}_t$ is a vector of shocks. $\mathbf{A}_0$ through $\mathbf{A}_3$ and $\mathbf{B}_0$ through $\mathbf{B}_3$ are matrices of parameters. The entries in $\mathbf{A}_i$ and $\mathbf{B}_j$ are all functions of the structural parameters, which we denote by vector $\boldsymbol{\theta}$. $\mathbf{A}_0$ and $\mathbf{B}_0$ are diagonal matrices, and $\mathbf{A}_2$ has zeros on its diagonal. $\mathbf{C}$ is a selection matrix that determines which state variables are subject to shocks.

For nonlinear DSGE models, (1) specifies the required structure of the equations. Equations can contain current values of state and control variables and can contain one-step-ahead values of state and control variables. All other terms, such as future values dated more than one period ahead or lagged values, are problematic.

For linearized DSGE models, (2) specifies the required structure for control variable equations, and (3) specifies the required structure for state variable equations. Per (2), a control can depend on only three types of terms—expected values of control variables in the next period, current values of other control variables, and current values of state variables. All other terms are problematic. Per (3), the values of state variables in the next period can depend on only four types of terms—expected values of control variables in the next period, current values of control variables, current values of state variables, and shocks. All other terms are problematic.

When written based on economic theory, the structural forms of many DSGE models include problematic terms that do not fit into the required form. These problematic terms are handled by defining a new state variable or a new control variable and rewriting the equations to have the required form. We define a new state variable when the problematic term involves an exogenous, also known as a predetermined, variable. We define a new control variable when the problematic term includes a new endogenous variable. The rewriting process replaces the variable in the problematic term with the new state or the new control variable.

Problematic terms usually violate one of the following four restrictions:

1. Equations for control variables may not contain shocks.
2. Equations may not contain lagged control variables.
3. Equations may not contain lagged state variables.
4. Equations may not contain expectations of control variables dated by more than one period ahead.

All of these restrictions can be overcome by adding a new state variable or a new control variable and rewriting the model so that it has the required form. Here are five examples of the four restrictions and solutions to each one.

1. If you have an equation for a control variable that contains a shock, make this shock a new state variable; see *Shocks to a control equation* below.
2. If you have an equation for a control variable that contains a lagged control variable, make the lagged control variable a new state variable; see *Including a lag of a control variable* below.
3. If you have an equation for a state variable that contains a lag of a state variable, make the lagged state variable a new state variable; see *Including a lag of a state variable* below.
4. If you have an equation for a control variable that contains an expectation of a control variable dated by more than one period ahead, make it a new control variable. We use a new control variable instead of a new state variable because the far-future expectation is endogenous, not predetermined. See *Including an expectation of a control dated by more than one period ahead* below for details.
5. If you have an equation for a control variable that contains a second-order lag of a control variable, make the second-order lagged control variable a new state variable. You will also need to create a new state variable for the first-order lag. See *Including a second-order lag of a control variable* below for details.

A nuanced issue not covered by restrictions 1–4 is that all the observed variables in a DSGE must be endogenous control variables. This structure is usually not a problem, because DSGE models are usually generated by theories that treat everything observed as endogenous. Occasionally, however, models include an observed variable that is exogenous. For example, a small, open economy might not be able to affect its real exchange rate. The solution for this problem is to model the observed variable as a control variable that is equal to an exogenous state variable. See *Including an observed exogenous variable* below for details.

For the most part, this entry discusses how to make models with problematic terms fit into the structure of (1) or (2) and (3). Note that correlated state variables already fit into this structure. This correlation is modeled by off-diagonal elements in $\mathbf{B_3}$, not by elements in $\mathbf{C}$. See [DSGE] **Intro 4g** for an example.

Shocks to a control equation

Sometimes, we observe only a control variable with error. Suppose that the equation for the control variable consumption is

$$c_t = E_t c_{t+1} - r_t + \epsilon_t \tag{4}$$

where c_t is consumption, r_t is the interest rate, and ϵ_t is a shock. Shocks to a control variable are not allowed, so ϵ_t is a problematic term.

The solution entails three steps. First, we define a new state variable $u_t = \epsilon_t$. We define a new state instead of a new control because the shock is exogenous. Second, we write a state equation for u_t. Recall that the state equation specifies tomorrow's state value in terms of today's state value and tomorrow's shock. The state equation for u_t is

$$u_{t+1} = \epsilon_{t+1}$$

Third, we rewrite (4), substituting u_t for ϵ_t. The consumption equation becomes

$$c_t = E_t c_{t+1} - r_t + u_t$$

which is in the required form. In the `dsge` command, this is expressed as

```
. dsge ...                    ///
       (c = F.c - r + u)      ///
       (F.u = 0, state)       ///
       ...
```

See [DSGE] **Intro 4a** for a complete example.

Including a lag of a control variable

Models that mix adaptive and rational expectations contain a lag of the control variable and the expected future value of the control variable. Suppose our model had the following equation for the control variable y_t,

$$y_t = \alpha y_{t-1} + (1-\alpha) E_t(y_{t+1}) - r_t \tag{4}$$

where y_t is output and r_t is the interest rate. The problematic term is αy_{t-1} because it involves a lag of the control variable.

The solution entails three steps. First, we define a new state variable $Ly_t = y_{t-1}$. We define a new state instead of new control because the lagged control is predetermined, which makes it exogenous. Second, we write its state equation as

$$Ly_{t+1} = y_t$$

Third, we rewrite (4), substituting Ly_t for y_{t-1},

$$y_t = \alpha Ly_t + (1-\alpha)E_t(y_{t+1}) - r_t$$

which is in the required form. To do this using the `dsge` command, we would type

```
. dsge ...                                          ///
       (y = {alpha}*Ly + (1-{alpha})*F.y - r)       ///
       (F.Ly = y, state noshock)                    ///
       ...
```

See [DSGE] **Intro 4b** for a complete example.

Including a lag of a state variable

Lagged state variables are handled in a manner analogous to lagged control variables. Suppose we have an equation for a state variable z_t that contains a lag of itself. For example, this occurs when z_t is a second-order autoregressive process instead of a first-order autoregressive process. Specifically, our model contains the equation

$$z_{t+1} = \rho_1 z_t + \rho_2 z_{t-1} + \epsilon_{t+1} \tag{5}$$

The problematic term is $\rho_2 z_{t-1}$ because it violates the form required by (2).

The solution entails three steps. First, we define a new state variable $Lz_t = z_{t-1}$. We define a new state instead of new control because the lagged state is exogenous. Second, we write its state equation as

$$Lz_{t+1} = z_t$$

Third, we rewrite (5), substituting Lz_t for z_{t-1},

$$z_{t+1} = \rho_1 z_t + \rho_2 Lz_t + \epsilon_{t+1}$$

which is in the required form. In the `dsge` command, we would type

```
. dsge ...                                  ///
       (F.z = {rho1}*z + {rho2}*Lz, state)  ///
       (F.Lz = z, state noshock)            ///
       ...
```

See [DSGE] **Intro 4c** for a complete example.

Including an expectation of a control dated by more than one period ahead

Many models include an expectation of a control variable dated by more than one period ahead. For example, suppose that the equation for the control variable consumption growth is

$$c_t = (1-h)w_t + hE_t c_{t+2} + r_t \tag{6}$$

where c_t is consumption growth, w_t is wage growth, and r_t is the interest rate. Only expectations of control variables dated one period ahead are allowed, so $E_t(c_{t+2})$ is a problematic term.

The solution entails three steps. First, we define a new control variable $Fc_t = E_t(c_{t+1})$. The new variable is a control instead of a state because $E_t(c_{t+1})$ is endogenous instead of exogenous. Second, we include this new control equation,

$$Fc_t = E_t(c_{t+1})$$

Third, we substitute Fc_{t+1} for c_{t+2} in (6). The equation becomes

$$c_t = (1-h)w_t + hE_t(Fc_{t+1}) + r_t$$

which is in the required form. Using the `dsge` command, we would type

```
. dsge ...                                  ///
       (c = (1-{h})*w + {h}*F.Fc + r)       ///
       (Fc = F.c, unobserved)               ///
       ...
```

See [DSGE] **Intro 4d** for a complete example.

Including a second-order lag of a control variable

Higher-order lags require defining more than one new state variable, as we illustrate here.

Equations for the state variable capital frequently involve "time to build", so that the control variable investment made in time t becomes available only as new capital several periods in the future. Suppose it takes three periods to convert investment x_t into capital. We denote current capital by k_t and write this three-period relationship as

$$k_{t+1} = (1-\delta)k_t + x_{t-2} \tag{7}$$

which specifies that the capital stock tomorrow is the sum of investment made two periods ago and the capital today that has not depreciated at rate δ.

The term x_{t-2} is problematic because it is a second-order lag of the state variable x_t, which violates the form required by (2).

The solution entails three steps. First, we define the new state variables $L2x_t = x_{t-2}$ and $Lx_t = x_{t-1}$. Second, we write their state equations as

$$L2x_{t+1} = Lx_t$$

and

$$Lx_{t+1} = x_t$$

Third, we rewrite (7) as

$$k_{t+1} = (1-\delta)k_t + L2x_t$$

With the `dsge` command, we could fit this model as

```
. dsge ...                                  ///
       (F.k = (1-{delta})*k + L2x)          ///
       (F.L2x = Lx, state noshock)          ///
       (F.Lx = x, state noshock)            ///
       ...
```

For a complete example, see [DSGE] **Intro 4e**.

In general, if the lag $t-k$ appears anywhere in any of your equations, you will need k new state equations.

Including an observed exogenous variable

Sometimes, we want to treat an observed variable as exogenous. Suppose we want to treat the changes in an observed exchange rate e_t as exogenous. The problem is that all the observed variables in a DSGE model must be modeled as endogenous control variables. The solution is to define a control variable that is equal to a state variable that models the exogenous process. So we could set the observed exchange rate e_t to equal the exchange rate state, es_t,

$$e_t = es_t$$

and specify, say, a first-order autoregressive process for es_t,

$$es_{t+1} = \rho_e es_t + \eta_{t+1}$$

To fit this type of model using the `dsge` command, we would type

```
. dsge ...                          ///
       (e = es)                     ///
       (F.es = {rhoe}*es, state)    ///
       ...
```

See [DSGE] **Intro 4f** for a complete example.

Also see

[DSGE] **Intro 2** — Learning the syntax

[DSGE] **Intro 3** — Classic DSGE examples

[DSGE] **dsge** — Linear dynamic stochastic general equilibrium models

Title

Intro 4a — Specifying a shock on a control variable

Description Remarks and examples Also see

Description

A shock to a control variable is problematic because it does not fit into the form of a structural model that is required to solve for the state-space form. This entry shows how to solve this problem by defining a new state variable and rewriting the equations.

Remarks and examples

Remarks are presented under the following headings:

The model

Equations (1)–(3) specify a model of consumption growth and growth in hours worked. The equation for consumption growth is subject to an additive shock.

$$c_t = (1-h)w_t + hE_tc_{t+1} + \epsilon_t \tag{1}$$
$$n_t = w_t - \gamma c_t \tag{2}$$
$$w_{t+1} = \rho w_t + \xi_{t+1} \tag{3}$$

Equation (1) specifies that consumption growth c_t is a linear combination of wage growth w_t, expected future consumption growth E_tc_{t+1}, and a consumption shock ϵ_t. Equation (2) specifies that the growth rate of hours worked n_t depends on wage growth and consumption growth. Equation (3) specifies an autoregressive process for wage growth. The structural parameter h controls the weights on wage growth and expected future consumption growth in the consumption equation.

One cannot solve the model in (1)–(3) for the state-space form because the shock ϵ_t is added to the control equation c_t in (1). Shocks to control variables must be reparameterized as new state variables so that the model can be solved for its state-space form. We define the shocks as new state variables instead of new control variables because the shocks are exogenous. We rewrite the model in (1)–(3) so that the additive shock is a new state variable in (4)–(7). Specifically, we define a new state variable z_t such that $z_t = \epsilon_t$ and write the model as

$$c_t = (1-h)w_t + hE_tc_{t+1} + z_t \tag{4}$$
$$n_t = w_t - \gamma c_t \tag{5}$$
$$w_{t+1} = \rho w_t + \xi_{t+1} \tag{6}$$
$$z_{t+1} = \epsilon_{t+1} \tag{7}$$

That we replaced ϵ_t with the new state variable z_t is the only difference between (4) and (1). Equations (5) and (6) are identical to their counterparts, (2) and (3). Equation (7) specifies the evolution of the new state variable z_t; it is similar to (6), except that it lacks an autoregressive component.

Parameter estimation

We estimate the parameters of the model in (4)–(7) using U.S. data on consumption growth and growth in hours worked. We specify `w` and `z` as state variables. We specify `c` and `n` as control variables. Both `c` and `n` are treated as observed. We specify that `w` and `z` are subject to shocks.

```
. use https://www.stata-press.com/data/r18/usmacro2
(Federal Reserve Economic Data - St. Louis Fed, 2017-01-15)
. dsge (c = (1-{h})*(w) +  {h}*F.c + z)
>      (n = w - {gamma}*c)
>      (F.w = {rho}*w, state)
>      (F.z = , state)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -2514.2527
Iteration 1:  Log likelihood =   -1264.15  (backed up)
Iteration 2:  Log likelihood = -1182.8885  (backed up)
Iteration 3:  Log likelihood = -1153.1143  (backed up)
Iteration 4:  Log likelihood = -1152.7297  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -1152.4548
Iteration 6:  Log likelihood = -1134.6813
Iteration 7:  Log likelihood = -1131.4432
Iteration 8:  Log likelihood =  -1131.283
Iteration 9:  Log likelihood = -1131.2826
Iteration 10: Log likelihood = -1131.2826
DSGE model
Sample: 1955q1 thru 2015q4                      Number of obs = 244
Log likelihood = -1131.2826
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
h	.7642505	.0360272	21.21	0.000	.6936386	.8348625
gamma	.2726181	.1059468	2.57	0.010	.0649663	.4802699
rho	.6545212	.0485627	13.48	0.000	.55934	.7497023
sd(e.w)	2.904762	.1958651			2.520873	3.28865
sd(e.z)	2.077219	.1400659			1.802695	2.351743

The estimate of `h` is 0.76. Being greater than 0.5, this indicates that consumption growth depends more strongly on expected future consumption growth than on current wage growth.

Also see

[DSGE] **Intro 2** — Learning the syntax

[DSGE] **Intro 4** — Writing a DSGE in a solvable form

Title

Intro 4b — Including a lag of a control variable

Description Remarks and examples Also see

Description

`dsge` and `dsgenl` do not allow lags of control variables to be included in the model. The structural form of the model that is required so that the model can be solved for its state-space form does not include lags of control variables. This entry shows how to fit a DSGE model that involves a lag of control variables by defining a new state variable and rewriting the equations.

Remarks and examples

Remarks are presented under the following headings:

A model with a lagged endogenous variable

The model in (1)–(5) is similar to many in monetary economics in that it includes inertia in the interest rate because the interest rate depends on its lagged value.

$$p_t = \beta E_t p_{t+1} + \kappa y_t \tag{1}$$
$$y_t = E_t y_{t+1} - (r_t - E_t p_{t+1} - \rho_z z_t) \tag{2}$$
$$r_t = \rho_r r_{t-1} + (1-\rho_r)\left(\frac{1}{\beta} p_t + u_t\right) \tag{3}$$
$$z_{t+1} = \rho_z z_t + \epsilon_{t+1} \tag{4}$$
$$u_{t+1} = \rho_u u_t + \xi_{t+1} \tag{5}$$

Equation (1) specifies the structural equation for inflation p_t. Inflation is a linear combination of expected future inflation $E_t(p_{t+1})$ and output growth y_t. Equation (2) specifies the structural equation for output growth. Output growth is a linear combination of expected future output growth $E_t(y_{t+1})$, the interest rate r_t, expected future inflation, and the state z_t. The state z_t is the first-order autoregressive process that drives output growth. Equation (3) specifies the structural equation for the interest rate. The interest rate depends on its own lagged value, the inflation rate, and the state u_t. The state u_t is the first-order autoregressive process that drives the interest rate. The control variables in this model are p_t, y_t, and r_t. The state variables are u_t and z_t.

The term involving r_{t-1} in (3) is problematic because the lag of a control variable does not fit into the structure required to solve the model for the state-space form. We accommodate this term by defining a new state variable Lr_t that equals r_{t-1} and replacing r_{t-1} in (3) with this new state variable. We define a new state instead of new control because the lagged control is predetermined, which makes it exogenous. These changes yield the model in (6)–(11).

$$p_t = \beta E_t p_{t+1} + \kappa y_t \tag{6}$$
$$y_t = E_t y_{t+1} - (r_t - E_t p_{t+1} - \rho_z z_t) \tag{7}$$
$$r_t = \rho_r Lr_t + (1-\rho_r)\left(\frac{1}{\beta}p_t + u_t\right) \tag{8}$$
$$Lr_{t+1} = r_t \tag{9}$$
$$z_{t+1} = \rho_z z_t + \epsilon_{t+1} \tag{10}$$
$$u_{t+1} = \rho_u u_t + \xi_{t+1} \tag{11}$$

Parameter estimation

We will estimate the parameters in the model in (6)–(11) using data on U.S. interest rate `r` and inflation `p`.

```
. use https://www.stata-press.com/data/r18/usmacro2
(Federal Reserve Economic Data - St. Louis Fed, 2017-01-15)
. dsge (p = {beta}*F.p + {kappa}*y)
>      (y = F.y - (r - f.p - {rhoz}*z), unobserved)
>      (r = {rhor}*lr + (1-{rhor})*((1/{beta})*p + u))
>      (F.lr = r, state noshock)
>      (F.u = {rhou}*u, state)
>      (F.z = {rhoz}*z, state)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -158313.11
Iteration 1:  Log likelihood = -5780.6489  (backed up)
Iteration 2:  Log likelihood = -1028.4602  (backed up)
Iteration 3:  Log likelihood = -1000.8509  (backed up)
Iteration 4:  Log likelihood = -873.38042  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -855.55057  (not concave)
Iteration 6:  Log likelihood = -803.87107  (not concave)
Iteration 7:  Log likelihood = -783.64132  (not concave)
Iteration 8:  Log likelihood = -775.43954  (not concave)
Iteration 9:  Log likelihood = -770.93147
Iteration 10: Log likelihood = -764.35421
Iteration 11: Log likelihood = -754.42169
Iteration 12: Log likelihood = -754.11237
Iteration 13: Log likelihood = -753.10614
Iteration 14: Log likelihood = -753.07125
Iteration 15: Log likelihood = -753.07026
Iteration 16: Log likelihood = -753.07026
```

```
DSGE model
Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -753.07026
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.5026374	.0791864	6.35	0.000	.347435	.6578398
kappa	.176019	.0511049	3.44	0.001	.0758553	.2761827
rhoz	.9591408	.0181341	52.89	0.000	.9235986	.994683
rhor	-.0939027	.093966	-1.00	0.318	-.2780727	.0902673
rhou	.7094626	.0447808	15.84	0.000	.6216939	.7972313
sd(e.u)	2.324309	.3236223			1.690021	2.958597
sd(e.z)	.6111535	.108498			.3985014	.8238056

Ironically, the inertia term for the interest rate `rhor` was not needed in this model. The autoregressive state u_t is probably modeling both the inertia and the persistence in the shocks to the interest rate.

Also see

[DSGE] **Intro 2** — Learning the syntax

[DSGE] **Intro 4** — Writing a DSGE in a solvable form

Title

Intro 4c — Including a lag of a state variable

Description Remarks and examples Also see

Description

Lags of state variables are not allowed to be included in the structural models specified in the `dsge` and `dsgenl` commands. These lags do not fit into the form of a structural model that can be solved for the state-space form. However, this restriction does not prevent us from fitting models that involve the lag of a state variable. This entry shows how to define a new state variable and rewrite the equations in the required structural form.

Remarks and examples

Remarks are presented under the following headings:

A model with a lagged state variable

The model in (1)–(5) is a standard monetary model in which the state-driving output growth y_t is a second-order autoregressive AR(2) process instead of a first-order autoregressive process.

$$p_t = \beta E_t(p_{t+1}) + \kappa y_t \tag{1}$$
$$y_t = E_t(y_{t+1}) - \{r_t - E_t(p_{t+1}) - z_t\} \tag{2}$$
$$r_t = \frac{1}{\beta} p_t + u_t \tag{3}$$
$$z_{t+1} = \rho_{z1} z_t + \rho_{z2} z_{t-1} + \epsilon_{t+1} \tag{4}$$
$$u_{t+1} = \rho_u u_t + \xi_{t+1} \tag{5}$$

Equation (1) specifies the structural equation for inflation p_t. Inflation is a linear combination of expected future inflation $E_t(p_{t+1})$ and output growth y_t. Equation (2) specifies the structural equation for output growth. Output growth is a linear combination of expected future output growth $E_t(y_{t+1})$, the interest rate r_t, expected future inflation, and the state z_t. The state z_t is an AR(2) process that drives output growth. Equation (3) specifies the structural equation for the interest rate. The interest rate depends on the inflation rate and the state u_t. The state u_t is an AR(1) process that drives the interest rate. The control variables in this model are p_t, y_t, and r_t. The state variables are u_t and z_t.

The AR(2) term in (4) is a problematic term because a lag of a state variable does not fit into the structure required to solve the model for the state-space form. We accommodate this term by defining a new state variable Lz_t that equals z_{t-1} and replacing z_{t-1} in (4) with this new state variable. We define a new state instead of new control because the lagged state is exogenous. These changes yield the model in (6)–(11).

$$p_t = \beta E_t(p_{t+1}) + \kappa y_t \tag{6}$$
$$y_t = E_t(y_{t+1}) - \{r_t - E_t(p_{t+1}) - z_t\} \tag{7}$$
$$r_t = \frac{1}{\beta} p_t + u_t \tag{8}$$
$$z_{t+1} = \rho_{z1} z_t + \rho_{z2} L z_t + \epsilon_{t+1} \tag{9}$$
$$u_{t+1} = \rho_u u_t + \xi_{t+1} \tag{10}$$
$$L z_{t+1} = z_t \tag{11}$$

We will estimate the parameters in the model in (6)–(11).

Parameter estimation

We estimate the parameters in the model in (6)–(11) using data on U.S. interest rate r and inflation p.

```
. use https://www.stata-press.com/data/r18/usmacro2
(Federal Reserve Economic Data - St. Louis Fed, 2017-01-15)
. dsge (p    = {beta}*F.p + {kappa}*y)
>      (y    = F.y - (r - F.p - z), unobserved)
>      (r    = (1/{beta})*p + u)
>      (F.u  = {rhou}*u, state)
>      (F.z  = {rhoz1}*z + {rhoz2}*Lz, state)
>      (F.Lz = z, state noshock)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -9116.0985
Iteration 1:  Log likelihood = -997.95396  (backed up)
Iteration 2:  Log likelihood = -881.53897  (backed up)
Iteration 3:  Log likelihood = -836.47478  (backed up)
Iteration 4:  Log likelihood = -769.37111  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -768.80775  (not concave)
Iteration 6:  Log likelihood = -760.17757  (not concave)
Iteration 7:  Log likelihood = -756.00241  (not concave)
Iteration 8:  Log likelihood = -754.93963  (not concave)
Iteration 9:  Log likelihood = -754.42345
Iteration 10: Log likelihood = -753.67337
Iteration 11: Log likelihood = -753.13078
Iteration 12: Log likelihood = -753.07841
Iteration 13: Log likelihood = -753.07788
Iteration 14: Log likelihood = -753.07788
DSGE model
Sample: 1955q1 thru 2015q4                      Number of obs = 244
Log likelihood = -753.07788
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.5154869	.0770491	6.69	0.000	.3644734	.6665004
kappa	.1662432	.0468473	3.55	0.000	.0744242	.2580622
rhou	.6979872	.0452071	15.44	0.000	.6093829	.7865915
rhoz1	.6735503	.2666048	2.53	0.012	.1510145	1.196086
rhoz2	.2709982	.2564615	1.06	0.291	-.2316571	.7736536
sd(e.u)	2.315264	.2992174			1.728809	2.90172
sd(e.z)	.7720659	.1716228			.4356915	1.10844

Looking at the confidence interval on rhoz2, we find no evidence that the AR(2) term is needed in this model.

Also see

[DSGE] **Intro 2** — Learning the syntax

[DSGE] **Intro 4** — Writing a DSGE in a solvable form

Title

Intro 4d — Including an expectation dated by more than one period ahead

Description Remarks and examples Also see

Description

Expectations of control variables dated by more than one period ahead are not allowed in the equations specified in `dsge` and `dsgenl` because they do not appear in the required form of a structural model that can be solved for the state-space form. In this entry, we demonstrate how to fit models with these types of expectations by defining a new variable and rewriting our equations. Because the expectation dated more than one period ahead is endogenous, the new variable that we define is a new control variable.

Remarks and examples

Remarks are presented under the following headings:

The model

Equations (1)–(4) specify a model of consumption growth and growth in hours worked.

$$c_t = (1-h)w_t + hE_tc_{t+2} + r_t \tag{1}$$
$$n_t = w_t - \gamma c_t \tag{2}$$
$$w_{t+1} = \rho_w w_t + \xi_{t+1} \tag{3}$$
$$r_{t+1} = \rho_r r_t + \epsilon_{t+1} \tag{4}$$

Equation (1) specifies that consumption growth c_t is a linear combination of wage growth w_t, the expected value of consumption growth two periods ahead E_tc_{t+2}, and the interest rate r_t. Equation (2) specifies that the growth rate of hours worked n_t depends on wage growth and consumption growth. Equations (3) and (4) specify a first-order autoregressive process for wage growth and for the interest rate, respectively. The control variables are c_t and n_t, and the state variables are w_t and r_t.

The expected value of consumption growth two periods ahead in (1) is a problematic term because it does not fit into the structure required to solve for the state-space form. The structure requires that expectations be only of one-period ahead values. We accommodate this term by defining a new control variable Fc_t that equals $E_t(c_{t+1})$ and replacing c_{t+1} in (1) with this new control variable. We define a new control variable instead of a new state variable because the expected future consumption is endogenous instead of exogenous. These changes yield the model in (5)–(9).

$$c_t = (1-h)w_t + hE_t(Fc_{t+1}) + r_t \tag{5}$$
$$n_t = w_t - \gamma c_t \tag{6}$$
$$Fc_t = E_t(c_{t+1}) \tag{7}$$
$$w_{t+1} = \rho_w w_t + \xi_{t+1} \tag{8}$$
$$r_{t+1} = \rho_r r_t + \epsilon_{t+1} \tag{9}$$

New (7) defines the new control variable Fc_t as the expected value of next period's consumption. Equation (5) is (1) but with c_{t+2} replaced with the new control Fc_{t+1}.

There are now three control variables and only two shocks, so we treat the new control variable Fc_t as unobserved. There is a logic to this process. The one-step-ahead structure of state-space models makes it impossible to solve for terms like $E_t(c_{t+2})$. In contrast, it is possible to solve for terms like $E_t\{E_t(c_{t+1})\}$ as long as the unobserved $E_t(c_{t+1})$ is determined by another equation, which in this case is (7). In fact, this example illustrates a part of the recursive dynamic modeling approach at the heart of the DSGE approach to macroeconomics.

Parameter estimation

We estimate the parameters of the model in (5)–(9) using U.S. data on consumption growth and growth in hours worked.

```
. use https://www.stata-press.com/data/r18/usmacro2
(Federal Reserve Economic Data - St. Louis Fed, 2017-01-15)
. dsge (c   = (1-{h})*(w) + {h}*F.fc + r)
>      (n   = w - {gamma}*c)
>      (fc  = F.c, unobserved)
>      (F.w = {rho_w}*w, state)
>      (F.r = {rho_r}*r, state)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -2423.7325
Iteration 1:  Log likelihood = -1284.9295  (backed up)
Iteration 2:  Log likelihood = -1193.2234  (backed up)
Iteration 3:  Log likelihood = -1180.1787  (backed up)
Iteration 4:  Log likelihood = -1175.3563  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -1171.0676  (backed up)
Iteration 6:  Log likelihood = -1154.3937
Iteration 7:  Log likelihood = -1137.4783
Iteration 8:  Log likelihood = -1130.5841
Iteration 9:  Log likelihood =  -1129.996
Iteration 10: Log likelihood = -1129.9359
Iteration 11: Log likelihood = -1129.9357
Iteration 12: Log likelihood = -1129.9357
DSGE model
Sample: 1955q1 thru 2015q4                      Number of obs = 244
Log likelihood = -1129.9357
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
h	.6617911	.0427917	15.47	0.000	.5779209	.7456612
gamma	.273381	.1120734	2.44	0.015	.0537212	.4930408
rho_w	.654523	.0485628	13.48	0.000	.5593417	.7497043
rho_r	.1050387	.0638171	1.65	0.100	-.0200406	.230118
sd(e.w)	2.905807	.2023203			2.509267	3.302348
sd(e.r)	2.049915	.1437862			1.7681	2.331731

The estimate of `h` is greater than 0.5, so we conclude that slightly more of the unobserved wage state is allocated to the expected growth in consumption two periods ahead than to current consumption.

Also see

[DSGE] **Intro 2** — Learning the syntax

[DSGE] **Intro 4** — Writing a DSGE in a solvable form

Title

Intro 4e — Including a second-order lag of a control

Description Remarks and examples Also see

Description

Some DSGE models capture delayed effects by including a second-order lag of a control variable and excluding the first-order lag. The second-order lag is a problematic term that does not fit into the form required to solve a structural model for its state-space form. This entry shows how to solve this problem by defining new state variables and rewriting the equations.

Remarks and examples

Remarks are presented under the following headings:

The model

Consider a model in which changes in hours worked take two periods to adjust because next period's hours have already been budgeted. In this model, the second-order lag of changes in hours worked is included, and the first-order lag is excluded. Equations (1)–(4) specify such a model of growth in hours worked and of consumption growth.

$$n_t = b_1 n_{t-2} + w_t - \gamma c_t \tag{1}$$
$$c_t = (1-h) w_t + h E_t c_{t+1} + r_t \tag{2}$$
$$w_{t+1} = \rho w_t + \xi_{t+1} \tag{3}$$
$$r_{t+1} = \epsilon_{t+1} \tag{4}$$

Equation (1) specifies that the growth rate of hours worked n_t depends on a second-order lag of itself, wage growth w_t, and consumption growth c_t. Equation (2) specifies that consumption growth is a linear combination of wage growth, expected future consumption growth $E_t c_{t+1}$, and the interest rate r_t. Equation (3) specifies an autoregressive process for wage growth. Equation (4) specifies that interest rate is just a shock. The control variables are n_t and c_t. The state variables are w_t and r_t.

One cannot solve the model in (1)–(4) for the state-space form because the problematic term $b_1 n_{t-2}$ does not fit into the required form. To accommodate this term, we define two new state variables, one for n_{t-1} and one for n_{t-2}. We define new state variables instead of new control variables because lags of the control are predetermined and thus exogenous. The model with new state variables is

$$n_t = b_1 L2n_t + w_t - \gamma c_t \tag{5}$$
$$c_t = (1-h)w_t + hE_t c_{t+1} + r_t \tag{6}$$
$$w_{t+1} = \rho w_t + \xi_{t+1} \tag{7}$$
$$r_{t+1} = \epsilon_{t+1} \tag{8}$$
$$Ln_{t+1} = n_t \tag{9}$$
$$L2n_{t+1} = Ln_t \tag{10}$$

Equation (9) defines the new state for n_{t-1}, and (10) defines $L2n_t$ to be the new state for n_{t-2}. The $L2n_t$ in (5) replaces n_{t-2} in (1).

Parameter estimation

We specify `n` and `c` as observed control equations. We specify `w`, `r`, `Ln`, and `L2n` as state equations. We specify that `w` and `r` are subject to shocks; the new states to accommodate n_{t-2} are not subject to shocks.

```
. use https://www.stata-press.com/data/r18/usmacro2
(Federal Reserve Economic Data - St. Louis Fed, 2017-01-15)
. dsge (n = {b1}*L2n + w - {gamma}*c)
>      (c = (1-{h})*w + {h}*F.c + r)
>      (F.w = {rho}*w, state)
>      (F.r = , state)
>      (F.L2n = Ln, state noshock)
>      (F.Ln = n, state noshock)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -2325.1996
Iteration 1:  Log likelihood = -1277.0146  (backed up)
Iteration 2:  Log likelihood = -1193.4512  (backed up)
Iteration 3:  Log likelihood = -1189.3181  (backed up)
Iteration 4:  Log likelihood = -1188.2629  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -1187.9872  (backed up)
Iteration 6:  Log likelihood =  -1147.018
Iteration 7:  Log likelihood = -1131.4022
Iteration 8:  Log likelihood = -1129.0383
Iteration 9:  Log likelihood = -1129.0181
Iteration 10: Log likelihood = -1129.0181
DSGE model
Sample: 1955q1 thru 2015q4                      Number of obs = 244
Log likelihood = -1129.0181
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
b1	.132084	.0608727	2.17	0.030	.0127758	.2513922
gamma	.3609224	.1298382	2.78	0.005	.1064442	.6154007
h	.7238124	.0406724	17.80	0.000	.6440959	.8035289
rho	.6177973	.0533568	11.58	0.000	.5132199	.7223746
sd(e.w)	3.033795	.2423843			2.55873	3.508859
sd(e.r)	1.970291	.1574526			1.66169	2.278893

Looking at the confidence interval for `b1`, we conclude that the second-order lag of hours' growth impacts current hours' growth.

Also see

[DSGE] **Intro 2** — Learning the syntax

[DSGE] **Intro 4** — Writing a DSGE in a solvable form

Title

Intro 4f — Including an observed exogenous variable

Description Remarks and examples Also see

Description

All the observed variables in a DSGE model must be modeled as endogenous control variables. This requirement implies that there is no reduced form for the endogenous variables as a function of observed exogenous variables. Any variable that is theoretically exogenous must be modeled.

Mechanically, the solution is to define a control variable that is equal to a state variable that models the exogenous process. We clarify this issue by discussing a model in which the exchange rate is theoretically exogenous.

Remarks and examples

Remarks are presented under the following headings:

The model

We model an economy in which the growth rate of the trade-weighted exchange rate is exogenous and in which it affects inflation. Henceforth, we call the trade-weighted exchange rate just the exchange rate. We begin by adding the growth rate of the exchange rate to the inflation equation in the New Keynesian model of [DSGE] **Intro 1**.

$$x_t = E_t(x_{t+1}) - \{r_t - E_t(p_{t+1}) - g_t\} \tag{1}$$
$$r_t = \frac{1}{\beta}p_t + u_t \tag{2}$$
$$p_t = \beta E_t(p_{t+1}) + \kappa x_t + \psi e_t \tag{3}$$
$$u_{t+1} = \rho_u u_t + \epsilon_{t+1} \tag{4}$$
$$g_{t+1} = \rho_g g_t + \xi_{t+1} \tag{5}$$

x_t is the output gap, which is modeled as an unobserved control variable. r_t is the interest rate, which is modeled as an observed control variable. p_t is the inflation rate, which is modeled as an observed control variable. g_t and u_t are first-order autoregressive state variables. e_t is the growth rate of the exchange rate, which we want to model as an observed exogenous variable.

The model in (1)–(5) cannot be solved because there is no equation for how the observed e_t evolves over time. A first-order autoregressive process [AR(1)] is a standard approximation to an exogenous variable, like the growth in exchange rate. We complete the model by adding an AR(1) for the unobserved state variable es_t and an equation linking the unobserved es_t to the observed e_t.

$$x_t = E_t(x_{t+1}) - \{r_t - E_t(p_{t+1}) - g_t\} \tag{6}$$

$$r_t = \frac{1}{\beta} p_t + u_t \tag{7}$$

$$p_t = \beta E_t(p_{t+1}) + \kappa x_t + \psi es_t \tag{8}$$

$$e_t = es_t \tag{8a}$$

$$u_{t+1} = \rho_u u_t + \epsilon_{t+1} \tag{9}$$

$$g_{t+1} = \rho_g g_t + \xi_{t+1} \tag{10}$$

$$es_{t+1} = \rho_e es_t + \eta_{t+1} \tag{11}$$

New (8a) specifies how the unobserved state es_t is transformed into the observed control variable e_t. Equation (11) specifies that the unobserved state es_t is an AR(1) process. Other equations are unchanged.

Parameter estimation

We fit this model using data on the U.S. interest rate `r`, inflation rate `p`, and the growth rate of the exchange rate `e`. The equation `(e = es)` links the observed variable `e` to the unobserved state variable `es`.

```
. use https://www.stata-press.com/data/r18/usmacro2
(Federal Reserve Economic Data - St. Louis Fed, 2017-01-15)
. dsge (x = F.x - (r - F.p - g), unobserved)
>      (r = 1/{beta}*p + u)
>      (p = {beta}*F.p + {kappa}*x + {psi}*es)
>      (e = es)
>      (F.u = {rho_u}*u, state)
>      (F.g = {rho_g}*g, state)
>      (F.es = {rho_e}*es, state)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -24249.537
Iteration 1:  Log likelihood = -8517.9588  (backed up)
Iteration 2:  Log likelihood = -1808.8075  (backed up)
Iteration 3:  Log likelihood = -1625.0334  (backed up)
Iteration 4:  Log likelihood = -1600.0843  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -1557.8925  (not concave)
Iteration 6:  Log likelihood = -1512.6127  (not concave)
Iteration 7:  Log likelihood = -1481.3062  (not concave)
Iteration 8:  Log likelihood =  -1276.109
Iteration 9:  Log likelihood = -1273.1796  (backed up)
Iteration 10: Log likelihood = -1202.6998  (not concave)
Iteration 11: Log likelihood = -1180.3345  (not concave)
Iteration 12: Log likelihood = -1177.3994  (not concave)
Iteration 13: Log likelihood = -1176.7022
Iteration 14: Log likelihood = -1175.4734
Iteration 15: Log likelihood = -1172.8554
Iteration 16: Log likelihood = -1172.1431
Iteration 17: Log likelihood = -1172.0483
Iteration 18: Log likelihood = -1172.0364
Iteration 19: Log likelihood = -1172.0364
DSGE model
Sample: 1973q2 thru 2015q4                              Number of obs = 171
Log likelihood = -1172.0364
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.5106726	.102212	5.00	0.000	.3103408	.7110044
kappa	.0901559	.0334259	2.70	0.007	.0246424	.1556695
psi	.01373	.0037684	3.64	0.000	.006344	.021116
rho_u	.7789342	.0473941	16.44	0.000	.6860435	.871825
rho_g	.9597848	.0206192	46.55	0.000	.919372	1.000198
rho_e	.2372213	.0742225	3.20	0.001	.0917479	.3826948
sd(e.u)	2.206083	.3465985			1.526763	2.885404
sd(e.g)	.7039306	.1545726			.4009738	1.006887
sd(e.es)	10.48932	.5671985			9.377629	11.60101

From the estimates for `psi`, it appears that the growth of the exchange rate impacts inflation.

Also see

[DSGE] **Intro 2** — Learning the syntax

[DSGE] **Intro 4** — Writing a DSGE in a solvable form

Title

Intro 4g — Correlated state variables

Description Remarks and examples Also see

Description

Many models include correlated state variables. We illustrate how to specify correlated state variables in a model of output growth y_t and inflation p_t.

Remarks and examples

Remarks are presented under the following headings:

The model

We model output growth y_t and inflation p_t as functions of domestic and international factors. The domestic factor g_t that drives output growth is a first-order autoregressive process that is also affected by the international factor z_t. The international factor that drives inflation is a simple first-order autoregressive process.

Mathematically, the model is

$$y_t = E_t y_{t+1} + \alpha p_t + g_t \tag{1}$$
$$p_t = z_t \tag{2}$$
$$g_{t+1} = \rho_g g_t + \rho_{gz} z_t + \xi_{t+1} \tag{3}$$
$$z_{t+1} = \rho_z z_t + \epsilon_{t+1} \tag{4}$$

Equation (1) specifies that output growth depends on expected future output growth $E_t y_{t+1}$, inflation p_t, and the domestic factor g_t. This equation has the form of an aggregate demand curve, and the parameter α is referred to as the slope of the aggregate demand curve. It should be negative. Equation (2) specifies that inflation is entirely driven by the international factor z_t. Equations (3) and (4) specify that the factors g_t and z_t follow a first-order vector autoregressive process with parameters ρ_g, ρ_{gz}, and ρ_z and with shocks ξ_{t+1} and ϵ_{t+1}. The factors z_t and g_t are the state variables, and p_t and y_t are the observed control variables.

Parameter estimation

From the U.S. macroeconomic data, we use the GDP-growth data in y and the inflation data in p and estimate the parameters of this model.

```
. dsge (y = F.y + {alpha}*p + g)
>      (p = z)
>      (F.g = {rho_g}*g + {rho_gz}*z, state)
>      (F.z = {rho_z}*z, state)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -2106.7245
Iteration 1:  Log likelihood = -1563.7386  (backed up)
Iteration 2:  Log likelihood = -1363.6417  (backed up)
Iteration 3:  Log likelihood = -1289.3079  (backed up)
Iteration 4:  Log likelihood = -1274.5732  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -1175.4471  (not concave)
Iteration 6:  Log likelihood = -1121.4993  (not concave)
Iteration 7:  Log likelihood = -1111.7243  (not concave)
Iteration 8:  Log likelihood = -1104.6131  (not concave)
Iteration 9:  Log likelihood = -1098.6694  (not concave)
Iteration 10: Log likelihood = -1084.9168  (not concave)
Iteration 11: Log likelihood = -1074.0252  (not concave)
Iteration 12: Log likelihood =   -1067.33  (not concave)
Iteration 13: Log likelihood = -1061.7529  (not concave)
Iteration 14: Log likelihood = -1061.0535
Iteration 15: Log likelihood = -1055.5719  (not concave)
Iteration 16: Log likelihood = -1035.9769
Iteration 17: Log likelihood = -1032.6914  (not concave)
Iteration 18: Log likelihood = -1025.4879  (not concave)
Iteration 19: Log likelihood = -1022.4293
Iteration 20: Log likelihood = -1019.3986  (not concave)
Iteration 21: Log likelihood = -1018.1331
Iteration 22: Log likelihood = -1017.7495
Iteration 23: Log likelihood = -1017.3913
Iteration 24: Log likelihood = -1017.1958
Iteration 25: Log likelihood = -1017.1594
Iteration 26: Log likelihood = -1017.1592
Iteration 27: Log likelihood = -1017.1592

DSGE model

Sample: 1955q1 thru 2015q4                      Number of obs = 244
Log likelihood = -1017.1592
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
alpha	-.1130024	.1554398	-0.73	0.467	-.4176589	.1916541
rho_g	.3357768	.0610763	5.50	0.000	.2160694	.4554842
rho_gz	.0443504	.09013	0.49	0.623	-.1323012	.221002
rho_z	.8626564	.0319007	27.04	0.000	.8001322	.9251806
sd(e.g)	2.184806	.2241106			1.745557	2.624054
sd(e.z)	1.146947	.0519234			1.045179	1.248715

The slope of the aggregate demand curve, α, is estimated to be negative as we expected, but the confidence interval is wide and includes zero. The imprecision in rho_gz has caused imprecision in the estimate of α.

Also see

[DSGE] **Intro 2** — Learning the syntax

[DSGE] **Intro 4** — Writing a DSGE in a solvable form

Title

Intro 5 — Stability conditions

Description Remarks and examples References Also see

Description

We can estimate only the parameters of models that we can solve, and we can solve only models that satisfy a stability condition. This entry discusses this condition, how it affects estimation in practice, and how to find initial values that satisfy it.

Remarks and examples

Remarks are presented under the following headings:

Why we care about stability

DSGE models are dynamic systems subject to random shocks. DSGE models that do not spiral out of control or converge to a single point when shocked are said to be "saddle-path stable". We can solve only saddle-path stable DSGE models, and we can estimate only the parameters of models we can solve.

The parameter values determine whether a DSGE model is saddle-path stable. Most DSGE models are saddle-path stable for some parameter values and not for other values. As we discussed in *Structural and reduced forms of DSGE models* in [DSGE] **Intro 1**, the structural form of a linear DSGE model is

$$\mathbf{A}_0\mathbf{y}_t = \mathbf{A}_1 E_t(\mathbf{y}_{t+1}) + \mathbf{A}_2\mathbf{y}_t + \mathbf{A}_3\mathbf{x}_t \tag{1}$$

$$\mathbf{B}_0\mathbf{x}_{t+1} = \mathbf{B}_1 E_t(\mathbf{y}_{t+1}) + \mathbf{B}_2\mathbf{y}_t + \mathbf{B}_3\mathbf{x}_t + \mathbf{C}\boldsymbol{\epsilon}_{t+1} \tag{2}$$

Given this structural form, the values in the $\mathbf{A}_i$ and $\mathbf{B}_j$ matrices determine saddle-path stability. For nonlinear DSGE models, saddle-path stability is assessed on the linear approximation to the nonlinear model at the steady state.

In the process of solving the structural form in (1) and (2) for the state space, the Klein (2000) solver computes the generalized eigenvalues of a matrix formed from the values in the $\mathbf{A}_i$ and $\mathbf{B}_j$ matrices. An eigenvalue is said to be stable when its absolute value is less than 1. The model is saddle-path stable when the number of stable eigenvalues equals the number of states in the model.

Once the maximization algorithm gets started, saddle-path stability usually is not an issue. Problems with saddle-path stability usually occur when trying to get the maximization process started. The initial values for the maximization algorithm must imply saddle-path stability because we cannot solve a DSGE model for parameter values that do not imply saddle-path stability.

`dsge` and `dsgenl` use a series of small positive numbers as the default initial values for the structural parameters that appear in the $\mathbf{A}$ and $\mathbf{B}$ matrices. If you know that your model is not saddle-path stable when a parameter, say, β, is set to 0.1, you should use the `from()` option to specify an initial value that does imply saddle-path stability. Alternatively, you could reparameterize the model so that $\beta = 0.1$ does imply saddle-path stability, but this solution tends to be more work than changing the initial value.

In rare cases, when the true parameters are close to values that are not saddle-path stable, the maximum likelihood estimator will suffer from convergence problems. Better initial values can help find the solution in these cases.

What if the initial values are not saddle-path stable?

If the initial values do not imply a saddle-path stable solution, you will see

```
. dsge ...
model is not saddle-path stable at current parameter values
    The number of stable eigenvalues is greater than the number of state
    variables.
  (output omitted)
```

or

```
. dsge ...
model is not saddle-path stable at current parameter values
    The number of stable eigenvalues is less than the number of state
    variables.
  (output omitted)
```

In either case, you should specify the `solve` and `noidencheck` options and subsequently use `estat stable` to find the problem. `solve` instructs `dsge` and `dsgenl` to only solve the model and not proceed with estimation, and `noidencheck` suppresses the check of whether the parameters are identified. `estat stable` will display the eigenvalues so that you can determine which ones are saddle-path stable and which ones are not. The trick is to change the starting values so that the number of stable eigenvalues equals the number of states in the model. Once you have determined which parameters are problematic, you can use the `from()` option to change the initial values to ones that do not have this problem.

Textbooks like those by Canova (2007), DeJong and Dave (2011), Ljungqvist and Sargent (2018), and Woodford (2003) are full of examples in which some structural parameters must be either greater than 1 or less than 1 for the model to be saddle-path stable. Similarly, some parameters must be either greater than or less than 0 for the model to be saddle-path stable. Moving one structural parameter over one of these boundaries and looking at how the change affects the number of stable eigenvalues can help find a vector of parameters that yields a saddle-path stable solution.

That the model that you are trying to fit is likely related to other models in the literature or in the textbooks suggests another strategy. Any such references probably contain some discussion of the set of values that yield saddle-path stable solutions.

In especially difficult cases, you can use the strategy of solving the problem for a smaller model and building back up. You temporarily remove equations from the model, find a parameter vector that yields a saddle-path stable solution for the smaller model, and then progressively add back equations as you find initial values that yield saddle-path stable solutions.

▷ Example 1: Finding saddle-path stable initial values

Equations (1)–(6) model the observed control variable (inflation) p_t, the unobserved control variable (output gap) y_t, and the observed control variable (interest rate) r_t as functions of the states u_t and z_t. Lz_{t+1} is an auxiliary state that allows z_t to be a second-order process instead of a first-order process.

$$p_t = (1/\gamma)E_t(p_{t+1}) + \kappa y_t \tag{3}$$
$$y_t = E_t(y_{t+1}) - \{r_t - E_t(p_{t+1}) - z_t\} \tag{4}$$
$$r_t = \gamma p_t + u_t \tag{5}$$
$$u_{t+1} = \rho_u u_t + \xi_{t+1} \tag{6}$$
$$z_{t+1} = \rho_{z1} z_t + \rho_{z2} L z_t + \epsilon_{t+1} \tag{7}$$
$$Lz_{t+1} = z_t \tag{8}$$

We try to fit this model using default starting values below.

```
. use https://www.stata-press.com/data/r18/usmacro2
(Federal Reserve Economic Data - St. Louis Fed, 2017-01-15)
. dsge (p    = (1/{gamma})*F.p + {kappa}*y)
>      (y    = F.y - (r - F.p - z), unobserved)
>      (r    = {gamma}*p + u)
>      (F.u  = {rho_u}*u, state)
>      (F.z  = {rho_z1}*z + {rho_z2}*Lz, state)
>      (F.Lz = z, state noshock)
model is not saddle-path stable at current parameter values
    The number of stable eigenvalues is greater than the number of state
    variables.
r(498);
```

The default initial values specify a model that is not saddle-path stable. We specify options `solve` and `noidencheck`, which cause the command to attempt only to solve the model and to skip the identification check.

```
. dsge (p    = (1/{gamma})*F.p + {kappa}*y)
>      (y    = F.y - (r - F.p - z), unobserved)
>      (r    = {gamma}*p + u)
>      (F.u  = {rho_u}*u, state)
>      (F.z  = {rho_z1}*z + {rho_z2}*Lz, state)
>      (F.Lz = z, state noshock),
>      solve noidencheck
DSGE model
Sample: 1955q1 thru 2015q4                          Number of obs = 244
Log likelihood = .
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
gamma	.21	.	.	.	.	.
kappa	.22	.	.	.	.	.
rho_u	.23	.	.	.	.	.
rho_z1	.24	.	.	.	.	.
rho_z2	.25	.	.	.	.	.
sd(e.u)	1	.			.	.
sd(e.z)	1	.			.	.

```
Note: Skipped identification check.
Warning: Model cannot be solved at current parameter values. Current values
         imply a model that is not saddle-path stable.
```

`dsge` displays the initial values used in the attempted solution, and it displays a warning message indicating that the model cannot be solved at these values because they imply a model that is not saddle-path stable.

We use `estat stable` to look at the eigenvalues implied by the initial values.

```
. estat stable
Stability results
```

	Eigenvalues
stable	-.3942
stable	.6342
stable	.21
stable	.23
unstable	2.605e+16
unstable	1.046

```
The process is not saddle-path stable.
The process is saddle-path stable when there are 3 stable eigenvalues and 3
    unstable eigenvalues.
```

The model is saddle-path stable when the number of stable eigenvalues equals the number of states in the model. The initial values yield four stable eigenvalues, but there are only three states in the model. So the model is not saddle-path stable at these values.

The initial value for `gamma` in the output from the failed solution attempt was 0.21. From (3), we see that the coefficient on expected future inflation is greater than one when `gamma` is less than one. We suspect that this relationship could be causing the lack of saddle-path stability. Below, we use option `from()` to specify 1.2 as an initial value for `gamma` and attempt to solve the model.

```
. dsge (p    = (1/{gamma})*F.p + {kappa}*y)
>      (y    = F.y - (r - F.p - z), unobserved)
>      (r    = {gamma}*p + u)
>      (F.u  = {rho_u}*u, state)
>      (F.z  = {rho_z1}*z + {rho_z2}*Lz, state)
>      (F.Lz = z, state noshock),
>      solve noidencheck from(gamma=1.2)
DSGE model
Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -1504.7564
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
gamma	1.2	.	.	.	.	.
kappa	.22	.	.	.	.	.
rho_u	.23	.	.	.	.	.
rho_z1	.24	.	.	.	.	.
rho_z2	.25	.	.	.	.	.
sd(e.u)	1	.			.	.
sd(e.z)	1	.			.	.

```
Note: Skipped identification check.
Note: Model solved at specified parameters.
```

There is no warning message, so the parameter values yielded a saddle-path stable solution. As a final illustration of this point, we display the eigenvalues below.

```
. estat stable

Stability results
```

	Eigenvalues
stable	-.3942
stable	.6342
stable	.23
unstable	5.380e+16
unstable	1.2
unstable	1.264

```
The process is saddle-path stable.
```

We could now proceed with estimation.

◁

References

Canova, F. 2007. *Methods for Applied Macroeconomic Research*. Princeton, NJ: Princeton University Press.

DeJong, D. N., and C. Dave. 2011. *Structural Macroeconometrics*. 2nd ed. Princeton, NJ: Princeton University Press.

Klein, P. 2000. Using the generalized Schur form to solve a multivariate linear rational expectations model. *Journal of Economic Dynamics and Control* 24: 1405–1423. https://doi.org/10.1016/S0165-1889(99)00045-7.

Ljungqvist, L., and T. J. Sargent. 2018. *Recursive Macroeconomic Theory*. 4th ed. Cambridge, MA: MIT Press.

Woodford, M. 2003. *Interest and Prices: Foundations of a Theory of Monetary Policy*. Princeton, NJ: Princeton University Press.

Also see

[DSGE] **Intro 6** — Identification

[DSGE] **Intro 7** — Convergence problems

[DSGE] **estat stable** — Check stability of system

Title

Intro 6 — Identification

Description Remarks and examples Reference Also see

Description

We can estimate only the parameters of a DSGE model when the parameters are identified. We discuss how to find the identification problems and how to fix them.

Remarks and examples

The parameters that maximize the likelihood function are the maximum likelihood (ML) estimates of ML estimators like the one in `dsge`. When there is a unique vector of parameters that maximizes the likelihood function, the parameters are identified and we can estimate them. When there is more than one parameter vector that produces the same maximum value of the likelihood function, the parameters are not identified, and we cannot estimate them because there is no way to choose among the sets of equally best parameters.

`dsge`, `dsgenl`, and all other commands that implement ML estimators check for identification at the solution by checking that the matrix of second derivatives, known as the Hessian, is full rank at the maximum. This test works only when the maximization algorithm arrives at a candidate maximum. In addition, it provides no information about which parameters are not identified.

For many models, when the parameters are not identified, the only output produced by a standard ML estimator is an ever growing series of "not concave" messages. This issue is especially relevant to DSGE models because it is easy to write down theoretical DSGE models whose parameters are not identified. `dsge` uses the Iskrev (2010) diagnostic to resolve this issue. The Iskrev diagnostic detects lack of parameter identification and reports which parameters are not identified at specific parameter values. By default, `dsge` performs the Iskrev diagnostic at the initial values and at the solution values. `dsgenl` performs the Iskrev diagnostic on the linear approximation to the nonlinear model at the steady state implied by the initial values and checks again at the solution values.

Instead of looking at the rank of the Hessian, the Iskrev (2010) diagnostic checks that all the parameters affect the autocovariances of the observed control variables. When there is a one-to-one relationship between the autocovariances for the observed control variables and the model parameters, the parameters are identified. Intuitively, this one-to-one relationship allows us to back out the model parameters from the autocovariances that we can always estimate from the data. When some of the parameters do not affect the autocovariances of the observed control variables, there is no one-to-one relationship, and the model parameters are not identified.

Simply adding restrictions to parameters can make the parameters of an unidentified model identified. This fact is frustrating but intuitive. When the parameters are not identified, they are not uniquely determined by the data generated by the true model. Only adding restrictions, either by defining constraints or by changing the equations, can identify the parameters.

▷ Example 1: A model with two unidentified parameters

Consider the following model of inflation p_t, output growth y_t, and the interest rate r_t.

$$
\begin{aligned}
p_t &= \beta E_t(p_{t+1}) + \kappa y_t \\
y_t &= E_t(y_{t+1}) - \gamma\{r_t - E_t(p_{t+1}) - \rho z_t\} \\
\beta r_t &= p_t + \beta u_t \\
z_{t+1} &= \rho z_t + \epsilon_{t+1} \\
u_{t+1} &= \delta u_t + \xi_{t+1}
\end{aligned}
$$

This model specifies how the observed control variables p_t and r_t depend on the unobserved control variable y_t, on the state variables z_t and u_t, and on the shocks ϵ_t and ξ_t.

Let's try to estimate the parameters of this model.

```
. use https://www.stata-press.com/data/r18/usmacro2
(Federal Reserve Economic Data - St. Louis Fed, 2017-01-15)
. dsge (p = {beta}*F.p + {kappa}*y)
>      (y = F.y -{gamma}*(r - F.p - {rhoz}*z), unobserved)
>      (r = (1/{beta})*p + u)
>      (F.u = {rhou}*u, state)
>      (F.z = {rhoz}*z, state)
identification failure at starting values
    Constrain some parameters or specify option noidencheck.  Likely source of
    identification failure: {kappa} {gamma}
r(498);
```

The error message indicates that `kappa` and `gamma` do not influence the autocovariances independently; they are linearly dependent. If a list of linearly dependent parameters is supplied, then you must constrain all but one of them prior to estimation. We constrain `gamma`.

```
. constraint 2 _b[gamma]=1
```

And we apply the constraint to the model:

```
. dsge (p = {beta}*F.p + {kappa}*y)
>      (y = F.y -{gamma}*(r - F.p - {rhoz}*z), unobserved)
>      (r = (1/{beta})*p + u)
>      (F.u = {rhou}*u, state)
>      (F.z = {rhoz}*z, state),
>      constraint(2)
(setting technique to bfgs)
Iteration 0:  Log likelihood =  -128443.1
Iteration 1:  Log likelihood = -3610.8865  (backed up)
Iteration 2:  Log likelihood = -1066.3053  (backed up)
Iteration 3:  Log likelihood =  -919.1875  (backed up)
Iteration 4:  Log likelihood = -815.49521  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -805.28492  (backed up)
Iteration 6:  Log likelihood = -790.04311
Iteration 7:  Log likelihood = -773.86016
Iteration 8:  Log likelihood =  -755.6535
Iteration 9:  Log likelihood = -753.96757
Iteration 10: Log likelihood = -753.57754
Iteration 11: Log likelihood = -753.57134
Iteration 12: Log likelihood = -753.57131
```

```
DSGE model
Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -753.57131
 ( 1)  [/structural]gamma = 1
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.5146687	.078349	6.57	0.000	.3611076	.6682298
kappa	.1659038	.0474064	3.50	0.000	.0729889	.2588187
gamma	1	(constrained)				
rhoz	.9545255	.0186424	51.20	0.000	.9179871	.9910639
rhou	.7005496	.0452601	15.48	0.000	.6118414	.7892577
sd(e.u)	2.318197	.3047421			1.720914	2.915481
sd(e.z)	.650711	.1123848			.4304409	.8709811

The remaining free parameters of the model are identified and can be estimated.

◁

Keep in mind that the Iskrev (2010) diagnostic is performed at specific parameter values. Sometimes, the parameters of a model cannot be identified at a vector of parameter values but can be identified at practically any other nearby vector. This problem arises when simple initial values cause terms to cancel out. If you suspect that the identification diagnostic is finding a problem because of initial values, you can either specify `noidencheck` or specify other seemingly random initial values.

Reference

Iskrev, N. 2010. Local identification in DSGE models. *Journal of Monetary Economics* 57: 189–202. https://doi.org/10.1016/j.jmoneco.2009.12.007.

Also see

[DSGE] **Intro 5** — Stability conditions

[DSGE] **Intro 7** — Convergence problems

Title

Intro 7 — Convergence problems

Description Remarks and examples Also see

Description

Fitting DSGE models is notoriously difficult. We discuss the causes of convergence problems and some solutions.

Remarks and examples

When the iteration process never gets started or when it never ends are two types of convergence problems. When it never gets started, you will see something like

```
. dsge ...
identification failure at starting values
  (output omitted)
```

This error occurs when the initial values specify a model that violates the stability conditions required for DSGE estimation; see [DSGE] **Intro 5** for a discussion of this issue and some solutions.

When the iteration process never ends, the cause is either that there is no unique solution or that the optimizer cannot find the unique solution. When there is no unique solution, the parameters are said to be not identified. In this case, you see `(not concave)` after each iteration, like

```
. dsge ...
  (output omitted)
Iteration 50: log likelihood = -337504.44 (not concave)
Iteration 51: log likelihood = -337503.52 (not concave)
Iteration 52: log likelihood = -337502.13 (not concave)
.
.
.
```

See [DSGE] **Intro 6** for a discussion of this issue and some solutions.

Even when the iterations get started and the parameters are identified, convergence problems are still common. In these cases, there are usually many parameters.

Changing the optimization technique is the simplest method to overcome a convergence problem, and it can be surprisingly effective. By default, `dsge` and `dsgenl` use the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm for five iterations before switching to a modified Newton–Raphson (NR) algorithm. The BFGS algorithm is especially effective at finding a maximum of the DSGE log-likelihood function from poor initial values. In some cases, simply specifying `technique(bfgs 200 nr)` can overcome the convergence problem.

The best way to overcome a convergence problem is to provide better initial values. The optimizer used by `dsge` and `dsgenl` will almost always converge when it is provided with good initial values and the parameters are identified. Specifying good initial values that come from theory or previous empirical work can solve your convergence problem. You can specify initial values inside the scalar substitutable expressions or by using the `from()` option; see examples 1 and 2 for details.

Sometimes, you have a convergence problem and do not have good initial values. One way to get initial values is to estimate a series of progressively less constrained models. This process usually produces convergence when the parameters are identified. See example 3 for details.

▷ Example 1: Specifying starting values in scalar substitutable expressions

Models with lagged state variables and other models that allow for more persistent processes can exhibit convergence problems. Equations (1)–(6) model the observed control variable (inflation) p_t, the unobserved control variable (output gap) y_t, and the observed control variable (interest rate) r_t as functions of the states u_t and z_t. Lz_{t+1} is an auxiliary state that allows z_t to be a second-order process instead of a first-order process.

$$p_t = \beta E_t(p_{t+1}) + \kappa y_t \tag{1}$$
$$y_t = E_t(y_{t+1}) - \{r_t - E_t(p_{t+1}) - z_t\} \tag{2}$$
$$r_t = \frac{1}{\beta} p_t + u_t \tag{3}$$
$$u_{t+1} = \rho_u u_t + \xi_{t+1} \tag{4}$$
$$z_{t+1} = \rho_{z1} z_t + \rho_{z2} L z_t + \epsilon_{t+1} \tag{5}$$
$$Lz_{t+1} = z_t \tag{6}$$

Suppose that previous empirical work suggests the initial values of $\beta = 0.5$, $\kappa = 0.2$, $\rho_u = 0.7$, $\rho_{z1} = 0.7$, and $\rho_{z2} = 0.2$. Inside a scalar substitutable expression, we can specify an initial value by typing {*parameter* = *value*}. We specify these initial values for the parameters in the scalar substitutable expressions in our model specification below.

```
. use https://www.stata-press.com/data/r18/usmacro2
(Federal Reserve Economic Data - St. Louis Fed, 2017-01-15)
. dsge (p    = {beta=.5}*F.p + {kappa=.2}*y)
>      (y    = F.y - (r - F.p - z), unobserved)
>      (r    = (1/{beta})*p + u)
>      (F.u  = {rho_u=.7}*u, state)
>      (F.z  = {rho_z1=.7}*z + {rho_z2=.2}*Lz, state)
>      (F.Lz = z, state noshock)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -1117.6457
Iteration 1:  Log likelihood = -936.74558  (backed up)
Iteration 2:  Log likelihood = -846.69924  (backed up)
Iteration 3:  Log likelihood = -835.21274  (backed up)
Iteration 4:  Log likelihood =  -782.8407  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -772.42535  (not concave)
Iteration 6:  Log likelihood = -754.37282
Iteration 7:  Log likelihood = -753.14308
Iteration 8:  Log likelihood = -753.07801
Iteration 9:  Log likelihood = -753.07788
Iteration 10: Log likelihood = -753.07788
```

```
DSGE model
Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -753.07788
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.5154873	.0770495	6.69	0.000	.3644731	.6665015
kappa	.1662426	.0468473	3.55	0.000	.0744236	.2580616
rho_u	.6979877	.0452071	15.44	0.000	.6093834	.786592
rho_z1	.6735472	.2665984	2.53	0.012	.1510239	1.196071
rho_z2	.2710011	.2564554	1.06	0.291	-.2316422	.7736444
sd(e.u)	2.315263	.2992181			1.728806	2.90172
sd(e.z)	.7720675	.1716204			.4356977	1.108437

Note that the output includes results for the standard deviations of the shocks, for which there are no scalar substitutable expressions. In example 2, we illustrate how to specify initial values for all the parameters using the `from()` option.

◁

▷ Example 2: Specifying starting values using from()

In example 1, we specified initial values inside the scalar substitutable expressions. You cannot specify initial values for the standard deviations of the shocks this way, because the shocks do not appear in any scalar substitutable expressions. We can specify starting values for any, or all, of the parameters using the `from()` option.

`from()` has several syntaxes, but the vector of starting values is most frequently used in DSGE applications. In this syntax, we specify a row vector containing the initial values $\beta = 0.5$, $\kappa = 0.2$, $\rho_u = 0.7$, $\rho_{z1} = 0.7$, $\rho_{z2} = 0.2$, $\sigma_{e.u} = 2.3$, and $\sigma_{e.z} = 0.7$. We begin by defining and listing the vector of initial values.

```
. matrix ivalues = (.5, .2, .7, .7, .2, 2.3, .7)
. matrix list ivalues
ivalues[1,7]
     c1   c2   c3   c4   c5   c6   c7
r1   .5   .2   .7   .7   .2  2.3   .7
```

Although we could specify and use the column names of the vector to assign elements of the vector to parameters, we use `from`'s suboption `copy` to make the assignment based on order of appearance in the vector. Specifying `from(, copy)` implies that the first element in the vector specifies the initial value for the first model parameter, the second element in the vector specifies the initial value for the second model parameter, and so on.

```
. dsge (p    = {beta}*F.p + {kappa}*y)
>      (y    = F.y - (r - F.p - z), unobserved)
>      (r    = (1/{beta})*p + u)
>      (F.u  = {rho_u}*u, state)
>      (F.z  = {rho_z1}*z + {rho_z2}*Lz, state)
>      (F.Lz = z, state noshock), from(ivalues, copy)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -769.09431
Iteration 1:  Log likelihood = -754.09564  (backed up)
Iteration 2:  Log likelihood = -753.70352  (backed up)
Iteration 3:  Log likelihood = -753.62091  (backed up)
Iteration 4:  Log likelihood = -753.53446  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -753.52617  (backed up)
Iteration 6:  Log likelihood = -753.08907
Iteration 7:  Log likelihood = -753.07789
Iteration 8:  Log likelihood = -753.07788
DSGE model
Sample: 1955q1 thru 2015q4                      Number of obs = 244
Log likelihood = -753.07788
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.5154875	.0770492	6.69	0.000	.3644738	.6665012
kappa	.1662423	.0468472	3.55	0.000	.0744236	.2580611
rho_u	.6979879	.0452071	15.44	0.000	.6093836	.7865922
rho_z1	.6735481	.2665998	2.53	0.012	.1510221	1.196074
rho_z2	.2710003	.2564567	1.06	0.291	-.2316456	.7736461
sd(e.u)	2.315262	.2992169			1.728808	2.901716
sd(e.z)	.7720669	.171621			.435696	1.108438

◁

▷ Example 3: Starting with a more constrained model to obtain convergence

Suppose that we wanted to estimate the parameters of the model in example 1 but that the only good initial value was that theory predicted that β should be about 0.5. Further suppose that we were having convergence problems fitting this model. In this case, one strategy would be to constrain β to 0.5 and to constrain κ to a small, nonzero value. The small value minimizes the interaction between p and y. Making the value nonzero can prevent problems when solving for the state-space form of the model. The remaining parameters determine the autoregressive processes for the states. Given that we suspect that the second-order lag is part of the problem, we also constrain it to a small, nonzero value. We try to estimate the remaining parameters in the output below.

```
. constraint define 1 _b[beta]   = 0.5
. constraint define 2 _b[kappa]  = 0.1
. constraint define 3 _b[rho_z2] = 0.01
. dsge (p    = {beta}*F.p + {kappa}*y)
>      (y    = F.y - (r - F.p - z), unobserved)
>      (r    = (1/{beta})*p + u)
>      (F.u  = {rho_u}*u, state)
>      (F.z  = {rho_z1}*z + {rho_z2}*Lz, state)
>      (F.Lz = z, state noshock), constraints(1 2 3)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -22446.965
Iteration 1:  Log likelihood = -2153.9862  (backed up)
Iteration 2:  Log likelihood = -1257.6437  (backed up)
Iteration 3:  Log likelihood = -1120.3354  (backed up)
Iteration 4:  Log likelihood = -1098.8062  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -1092.0433
Iteration 6:  Log likelihood = -795.47417
Iteration 7:  Log likelihood = -757.40637
Iteration 8:  Log likelihood =   -755.662
Iteration 9:  Log likelihood = -755.63027
Iteration 10: Log likelihood = -755.63025
DSGE model
Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -755.63025
 ( 1)  [/structural]beta = .5
 ( 2)  [/structural]kappa = .1
 ( 3)  [/structural]rho_z2 = .01
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.5	(constrained)				
kappa	.1	(constrained)				
rho_u	.7906612	.0102382	77.23	0.000	.7705947	.8107277
rho_z1	.9440872	.0188166	50.17	0.000	.9072073	.980967
rho_z2	.01	(constrained)				
sd(e.u)	2.391372	.1083319			2.179045	2.603698
sd(e.z)	.6970129	.0731968			.5535498	.840476

The estimator converged. We will want to use these estimates as initial values, so we put them into the matrix `b` and list `b` to confirm it has the desired values.

```
. matrix b = e(b)
. matrix list b
b[1,7]
     /structural:  /structural:  /structural:  /structural:  /structural:
            beta         kappa         rho_u        rho_z1        rho_z2
y1            .5            .1     .79066118     .94408716           .01
              /:            /:
         sd(e.u)       sd(e.z)
y1     2.3913715     .69701292
```

We use the previous estimates as initial values, drop the constraints, and try to estimate all the parameters.

```
. dsge (p    = {beta}*F.p + {kappa}*y)
>      (y    = F.y - (r - F.p - z), unobserved)
>      (r    = (1/{beta})*p + u)
>      (F.u  = {rho_u}*u, state)
>      (F.z  = {rho_z1}*z + {rho_z2}*Lz, state)
>      (F.Lz = z, state noshock), from(b, copy)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -755.63025
Iteration 1:  Log likelihood = -755.57974  (backed up)
Iteration 2:  Log likelihood =  -755.1865  (backed up)
Iteration 3:  Log likelihood = -754.62381  (backed up)
Iteration 4:  Log likelihood =  -754.5529  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -754.45439  (backed up)
Iteration 6:  Log likelihood = -753.40634
Iteration 7:  Log likelihood = -753.09883
Iteration 8:  Log likelihood = -753.07802
Iteration 9:  Log likelihood = -753.07788
Iteration 10: Log likelihood = -753.07788
DSGE model
Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -753.07788
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.5154873	.07705	6.69	0.000	.3644721	.6665024
kappa	.1662425	.0468474	3.55	0.000	.0744232	.2580617
rho_u	.6979879	.0452071	15.44	0.000	.6093835	.7865922
rho_z1	.6735472	.2666187	2.53	0.012	.1509841	1.19611
rho_z2	.2710012	.2564748	1.06	0.291	-.2316803	.7736826
sd(e.u)	2.315264	.2992199			1.728803	2.901724
sd(e.z)	.7720676	.1716278			.4356832	1.108452

The estimator converged.

In larger problems, more steps are sometimes required. In these steps, you use the previous estimates as initial values, but you drop only some of the constraints at each step.

◁

Also see

[DSGE] **Intro 5** — Stability conditions

[DSGE] **Intro 6** — Identification

Title

Intro 8 — Wald tests vary with nonlinear transforms

Description Remarks and examples References Also see

Description

After fitting a DSGE model, we often perform tests of structural parameters, and these tests often place nonlinear restrictions on the parameters. The values and rejection rates of a Wald test for different nonlinear expressions of the same null hypothesis are different. We illustrate this issue, show that likelihood-ratio (LR) tests do not have this problem, and illustrate that you can parameterize your model in terms of invertible transforms of each parameter.

Remarks and examples

Remarks are presented under the following headings:

Wald tests vary with nonlinear transforms

Performing a statistical test of whether a structural parameter in a DSGE has a specific value is one of the most frequent forms of inference after `dsge` and `dsgenl` estimation. The null hypothesis in one of these tests frequently places nonlinear restrictions on the underlying parameters. Two different nonlinear expressions of the same null hypothesis produce different Wald test statistics in finite samples and have different rejection rates. In other words, the Wald test is not invariant to nonlinear transforms of the null hypothesis. The LR test, on the other hand, is invariant to nonlinear transforms of the null hypothesis.

▷ Example 1: Different values from logically equivalent Wald tests

Equations (1)–(5) specify how the observed control variable inflation p_t, the unobserved control variable output growth y_t, and the observed control variable (interest rate) r_t depend on the states z_t and u_t, given the shocks ϵ_t and ξ_t.

$$p_t = \beta E_t(p_{t+1}) + \kappa y_t \tag{1}$$
$$y_t = E_t(y_{t+1}) - \{r_t - E_t(p_{t+1}) - \rho z_t\} \tag{2}$$
$$r_t = (1/\beta)p_t + u_t \tag{3}$$
$$z_{t+1} = \rho z_t + \epsilon_{t+1} \tag{4}$$
$$u_{t+1} = \delta u_t + \xi_{t+1} \tag{5}$$

We estimate the parameters of this model using the macroeconomic data for the United States in `usmacro2.dta`.

```
. use https://www.stata-press.com/data/r18/usmacro2
(Federal Reserve Economic Data - St. Louis Fed, 2017-01-15)
. dsge (p = {beta}*F.p + {kappa}*y)
>      (y = F.y - (r - F.p - {rhoz}*z), unobserved)
>      (r = (1/{beta})*p + u)
>      (F.u = {rhou}*u, state)
>      (F.z = {rhoz}*z, state)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -146218.64
Iteration 1:  Log likelihood = -5532.4212  (backed up)
Iteration 2:  Log likelihood = -1067.4665  (backed up)
Iteration 3:  Log likelihood = -938.92415  (backed up)
Iteration 4:  Log likelihood = -885.96401  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -880.81743  (not concave)
Iteration 6:  Log likelihood = -818.95369
Iteration 7:  Log likelihood = -787.30327
Iteration 8:  Log likelihood = -754.54306
Iteration 9:  Log likelihood = -753.62794
Iteration 10: Log likelihood = -753.57273
Iteration 11: Log likelihood = -753.57131
Iteration 12: Log likelihood = -753.57131
DSGE model
Sample: 1955q1 thru 2015q4                      Number of obs = 244
Log likelihood = -753.57131
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.5146674	.0783489	6.57	0.000	.3611065	.6682283
kappa	.1659057	.0474074	3.50	0.000	.072989	.2588224
rhoz	.9545256	.0186424	51.20	0.000	.9179872	.991064
rhou	.7005482	.0452604	15.48	0.000	.6118394	.789257
sd(e.u)	2.318202	.3047435			1.720916	2.915489
sd(e.z)	.6507117	.1123844			.4304423	.8709811

The interest rate equation shown in (3) links the nominal interest rate to the inflation rate. The coefficient on inflation is $1/\beta$. We test whether this parameter is 1.5, a common benchmark value in the literature.

```
. testnl 1/_b[beta] = 1.5
  (1)  1/_b[beta] = 1.5
               chi2(1) =        2.24
           Prob > chi2 =        0.1342
```

We do not reject the null hypothesis that $1/\beta$ is 1.5.

If we test the logically equivalent hypothesis that $\beta = 2/3$, the statistic and p-value change.

```
. test _b[beta] =2/3
 ( 1)  [/structural]beta = .6666667
           chi2(  1) =    3.76
         Prob > chi2 =    0.0524
```

The values of these two logically equivalent Wald tests differ because Wald tests are not invariant to nonlinear transformation. This issue is well known in the literature; see Gregory and Veall (1985) and Phillips and Park (1988) for details. In this example, the inference of failing to reject the null hypothesis remains the same when using a 5% significance level, but this is not true in general. Different formulations of Wald tests can lead to different inferences.

◁

LR tests do not vary with nonlinear transforms

▷ Example 2: LR tests are invariant to nonlinear transforms

We illustrate this feature by performing LR tests that $\beta = 2/3$ and that $1/\beta = 1.5$. The current estimates are those of the unconstrained model. We repeat these results and store them as `unconstrained`.

```
. dsge
DSGE model
Sample: 1955q1 thru 2015q4                          Number of obs = 244
Log likelihood = -753.57131
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.5146674	.0783489	6.57	0.000	.3611065	.6682283
kappa	.1659057	.0474074	3.50	0.000	.072989	.2588224
rhoz	.9545256	.0186424	51.20	0.000	.9179872	.991064
rhou	.7005482	.0452604	15.48	0.000	.6118394	.789257
sd(e.u)	2.318202	.3047435			1.720916	2.915489
sd(e.z)	.6507117	.1123844			.4304423	.8709811

```
. estimates store unconstrained
```

Now, we estimate the parameters of the constrained model in which $\beta = 2/3$, store the results as `constrained`, and perform an LR test of the null hypothesis that $\beta = 2/3$.

```
. constraint 1 _b[beta] = 2/3

. dsge (p = {beta}*F.p + {kappa}*y)
>      (y = F.y - (r - F.p - {rhoz}*z), unobserved)
>      (r = (1/{beta})*p + u)
>      (F.u = {rhou}*u, state)
>      (F.z = {rhoz}*z, state),
>      constraint(1)
(setting technique to bfgs)
Iteration 0:  Log likelihood =  -119695.1
Iteration 1:  Log likelihood =  -1425.592  (backed up)
Iteration 2:  Log likelihood = -984.57609  (backed up)
Iteration 3:  Log likelihood = -948.41524  (backed up)
Iteration 4:  Log likelihood = -945.83724  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -945.06881  (backed up)
Iteration 6:  Log likelihood = -760.71545
Iteration 7:  Log likelihood = -755.52634
Iteration 8:  Log likelihood = -755.11897
Iteration 9:  Log likelihood = -755.11007
Iteration 10: Log likelihood = -755.11003
DSGE model
Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -755.11003
 ( 1)  [/structural]beta = .6666667
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
beta	.6666667	(constrained)				
kappa	.1076811	.0276892	3.89	0.000	.0534113	.1619509
rhoz	.9538522	.0187789	50.79	0.000	.9170462	.9906581
rhou	.7214328	.0439669	16.41	0.000	.6352593	.8076063
sd(e.u)	1.915459	.0867103			1.74551	2.085408
sd(e.z)	.4936797	.080513			.3358771	.6514822

```
. estimates store constrained

. lrtest unconstrained constrained

Likelihood-ratio test
Assumption: constrained nested within unconstrained

 LR chi2(1) =   3.08
Prob > chi2 = 0.0794
```

Note that the value of the LR statistic is 3.08. We now illustrate an LR of the null hypothesis that $1/\beta = 1.5$ produces the same value.

We cannot impose nonlinear restrictions on parameters, so we must begin by reparameterizing the unconstrained model by replacing `{beta}` with `1/{beta}`. To avoid having `{beta}` mean two different things, we write the model in terms of `{gamma}=1/{beta}` and estimate the parameters:

```
. dsge (p = 1/{gamma}*F.p + {kappa}*y)
>      (y = F.y - (r - F.p - {rhoz}*z), unobserved)
>      (r = ({gamma})*p + u)
>      (F.u = {rhou}*u, state)
>      (F.z = {rhoz}*z, state),
>     from(gamma=2 kappa=0.15 rhou=0.75 rhoz=0.95)
(setting technique to bfgs)
Iteration 0:  Log likelihood = -1137.8808
Iteration 1:  Log likelihood = -1097.9283  (backed up)
Iteration 2:  Log likelihood = -1027.9554  (backed up)
Iteration 3:  Log likelihood = -801.19555  (backed up)
Iteration 4:  Log likelihood = -784.48041  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -763.19407  (not concave)
Iteration 6:  Log likelihood = -754.49971  (not concave)
Iteration 7:  Log likelihood = -754.08362
Iteration 8:  Log likelihood = -753.57362
Iteration 9:  Log likelihood = -753.57131
Iteration 10: Log likelihood = -753.57131
DSGE model
Sample: 1955q1 thru 2015q4                      Number of obs = 244
Log likelihood = -753.57131
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
gamma	1.943005	.2957867	6.57	0.000	1.363273	2.522736
kappa	.1659061	.0474073	3.50	0.000	.0729895	.2588226
rhoz	.9545256	.0186424	51.20	0.000	.9179872	.991064
rhou	.7005481	.0452604	15.48	0.000	.6118393	.7892568
sd(e.u)	2.318205	.3047433			1.720919	2.915491
sd(e.z)	.6507124	.1123842			.4304434	.8709813

```
. estimates store unconstrained2
```

The estimates of the parameters other than `gamma` and the value of the log likelihood are nearly the same as those for the `unconstrained` model. The value for `gamma` $= 1.94$ is the same as `1/beta` $= 1/0.514 = 1.95$. By tightening the convergence tolerance, we could make these values exactly the same. These values are nearly the same because this example is an instance of a general property of maximum likelihood estimators. Transforming a parameter by an invertible function does not change the log likelihood or the other parameter estimates. In other words, maximum likelihood estimators are invariant to invertible transformations of the parameters; see Casella and Berger (2002, 319) for details.

Having stored the estimates from the unconstrained model, we now estimate the parameters of the constrained model and store these results in `constrained2`.

```
. constraint 2 _b[gamma] = 1.5

. dsge (p = 1/{gamma}*F.p + {kappa}*y)
>      (y = F.y - (r - F.p - {rhoz}*z), unobserved)
>      (r = ({gamma})*p + u)
>      (F.u = {rhou}*u, state)
>      (F.z = {rhoz}*z, state),
>      constraint(2)
(setting technique to bfgs)
Iteration 0:  Log likelihood =  -119695.1
Iteration 1:  Log likelihood =  -1425.592  (backed up)
Iteration 2:  Log likelihood = -984.57609  (backed up)
Iteration 3:  Log likelihood = -948.41524  (backed up)
Iteration 4:  Log likelihood = -945.83724  (backed up)
(switching technique to nr)
Iteration 5:  Log likelihood = -945.06881  (backed up)
Iteration 6:  Log likelihood = -760.71545
Iteration 7:  Log likelihood = -755.52634
Iteration 8:  Log likelihood = -755.11897
Iteration 9:  Log likelihood = -755.11007
Iteration 10: Log likelihood = -755.11003

DSGE model

Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -755.11003
 ( 1)  [/structural]gamma = 1.5
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
gamma	1.5	(constrained)				
kappa	.1076811	.0276892	3.89	0.000	.0534113	.1619509
rhoz	.9538522	.0187789	50.79	0.000	.9170462	.9906581
rhou	.7214328	.0439669	16.41	0.000	.6352593	.8076063
sd(e.u)	1.915459	.0867103			1.74551	2.085408
sd(e.z)	.4936797	.080513			.3358771	.6514822

```
. estimates store constrained2
```

The estimates of the parameters other than `gamma` and the value of the log likelihood are the same as those for the `constrained` model. This is another instance of the invariance of the maximum likelihood estimator to invertible transformations of the parameters.

Having stored the log likelihoods from the constrained and unconstrained model, we now perform an LR of the null hypothesis that $\gamma = 1.5$.

```
. lrtest unconstrained2 constrained2

Likelihood-ratio test
Assumption: constrained2 nested within unconstrained2

 LR chi2(1) =   3.08
Prob > chi2 = 0.0794
```

The LR test statistic and its p-value are the same as those reported for the test against the null hypothesis that $\beta = 2/3$, which illustrates that LR tests are invariant to nonlinear transforms.

◁

References

Casella, G., and R. L. Berger. 2002. *Statistical Inference*. 2nd ed. Pacific Grove, CA: Duxbury.

Gregory, A. W., and M. R. Veall. 1985. Formulating Wald tests of nonlinear restrictions. *Econometrica* 53: 1465–1468. https://doi.org/10.2307/1913221.

Phillips, P. C. B., and J. Y. Park. 1988. On the formulation of Wald tests of nonlinear restrictions. *Econometrica* 56: 1065–1083. https://doi.org/10.2307/1911359.

Also see

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

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

[R] **lrtest** — Likelihood-ratio test after estimation

[R] **test** — Test linear hypotheses after estimation

Title

Intro 9 — Bayesian estimation

Description Remarks and examples Also see

Description

This entry introduces Bayesian estimation of DSGE models. We discuss the setup of such models, the specification of priors, diagnostics tests, and other statistical and computational details important to fit Bayesian models.

We assume that you are already familiar with the elements of a DSGE; see [DSGE] **Intro 1**. For a comprehensive discussion of Bayesian analysis and Bayesian methods using Stata, see [BAYES] **Intro** and [BAYES] **Bayesian commands**.

Remarks and examples

Remarks are presented under the following headings:

Introduction

Bayesian estimation provides an alternative to maximum likelihood estimation. Bayesian analysis combines a likelihood model for the data with a specification of prior knowledge of the distribution of parameters to arrive at a posterior distribution for the parameters. Incorporation of prior information in DSGE models takes several forms.

First, the prior incorporates information that would be difficult or impossible to specify through the likelihood, broadening the amount of information that can be incorporated into the model. The likelihood is typically specified in terms of aggregate equations. However, some information about the parameters can be drawn from microeconomic evidence. This evidence can be incorporated into estimation via the prior. Second, the prior incorporates logical bounds on values that a parameter might take. Some parameters represent shares, like a capital share of income or depreciation rate of capital, so that only the (0,1) interval is an appropriate value for the parameter. A prior that takes on positive value in (0,1) but has a zero probability elsewhere restricts the search to the desired region. Third, the prior can reflect information from theory, for instance, that a parameter should be positive or should lie above a critical threshold. All of this makes Bayesian estimation of DSGE models an appealing alternative to classical frequentist estimation.

Stata provides a full suite of commands for Bayesian estimation. See [BAYES] **Intro** for an introduction to Bayesian analysis and [BAYES] **Bayesian commands** for an introduction to Stata's implementation of Bayesian analysis. This entry focuses on Stata's implementation of Bayesian estimation for DSGE models.

Bayesian estimation of DSGE models is accomplished through use of the `bayes:` prefix. If your DSGE model is

```
. dsge (y = z) (F.z = {rho}*z, state)
```

then the corresponding Bayesian estimates can be obtained with

```
. bayes, prior({rho}, uniform(0,1)): dsge (y=z) (F.z={rho}*z, state)
```

The command has two parts: the DSGE model and the prior specification. The DSGE model is specified exactly as it is when estimating using maximum likelihood. Priors are specified with the `prior()` option of `bayes`. Most `bayes` commands come with default priors, intended to be uninformative; see *Remarks and examples* in [BAYES] **bayes**. However, in the case of DSGE models, default priors are only provided for standard deviation parameters. All other model parameters must be accompanied by an explicit prior. DSGE parameters are often logically bounded to a certain range by the model structure, such as the (0,1) interval or greater than some critical value, so that uninformative priors over the entire real line are inappropriate. Because DSGE parameters are model dependent, informative priors are readily available and in most cases necessary for estimates to be reasonable.

Principles of Bayesian DSGE estimation

An uninformative prior

We begin with an autoregressive model of order 1, written in its state-space form. This model is used to provide an introduction to the syntax of `bayes: dsge` and the specification of priors. See [TS] **sspace** for more information on the state-space form of an autoregressive model, and see [BAYES] **bayes: var** for a more general approach to the estimation of autoregression and vector autoregression models using Bayesian methods. We can estimate the parameters of this model by maximum likelihood using the `dsge` command (see [DSGE] **dsge**). We use output growth `y` as the dependent variable.

```
. use https://www.stata-press.com/data/r18/usmacro2
(Federal Reserve Economic Data - St. Louis Fed, 2017-01-15)
. dsge (y = z) (F.z = {rho}*z, state), nolog
DSGE model
Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -639.18399
```

y	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/structural						
rho	.3392504	.0608551	5.57	0.000	.2199765	.4585243
sd(e.z)	3.321503	.1503577			3.026808	3.616199

The first-order autocorrelation of output growth is 0.339, and the standard deviation of output shock is 3.32.

To estimate these parameters using Bayesian methods, specify a prior for model parameters, and use the `bayes:` prefix. Priors for most model parameters are required. Default priors are provided for the parameters representing the standard deviation of shocks. For a first example, we specify a flat prior for the autocorrelation parameter ρ in the range $(-1, 1)$. The command is

```
. bayes, prior({rho}, uniform(-1, 1)) : dsge (y = z) (F.z = {rho}*z, state)
```

Because Bayesian estimation is a stochastic procedure, we specify the random-number seed to ensure reproducibility.

```
. bayes, rseed(17) prior({rho}, uniform(-1,1)): dsge (y=z) (F.z={rho}*z, state)
note: initial parameter vector set to means of priors.
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  y ~ dsgell({rho},{sd(e.z)})
Priors:
      {rho} ~ uniform(-1,1)
  {sd(e.z)} ~ igamma(.01,.01)
------------------------------------------------------------------------------
Bayesian linear DSGE model                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
Sample: 1955q1 thru 2015q4                      Number of obs    =        244
                                                Acceptance rate  =      .2635
                                                Efficiency:  min =     .07655
                                                             avg =      .1075
Log marginal-likelihood =  -648.5502                         max =      .1385
------------------------------------------------------------------------------
                                                              Equal-tailed
             |      Mean   Std. dev.     MCSE     Median  [95% cred. interval]
-------------+----------------------------------------------------------------
         rho |   .3385195   .0611807   .001644   .3388822    .2251003    .465511
      sd(e.z)    3.33619   .1564792   .005656   3.330808    3.062057   3.666267
------------------------------------------------------------------------------
```

The model summary shows the likelihood model and the priors used for model parameters. The likelihood model is `dsgell()`, meaning the state-space form of the DSGE model. The prior for `{rho}` is $\text{uniform}(-1, 1)$. The prior for the standard deviation of the state variable, `{sd(e.z)}`, is by default an inverse-gamma prior with parameters (0.01, 0.01).

The estimation header reports the total number of MCMC draws, the length of the burn-in period, the number of MCMC draws used for estimation, the number of observations in the dataset, the acceptance rate, and the efficiency rates of the draws. The acceptance rate specifies the proportion of proposed parameter values accepted by the MCMC algorithm. For the Metropolis–Hastings algorithm used by Bayesian DSGE estimation, this number is rarely above 50% and is often below 30%. A low acceptance rate—say, below 10%—can indicate convergence problems.

The first two columns of the estimation table report the posterior mean and posterior standard deviation. Because the prior was uninformative, the posterior mean of the autoregressive parameter `{rho}`, 0.339, is the same as the maximum likelihood estimate of 0.339. The estimation table also reports Monte Carlo standard errors, medians, and equal-tailed credible intervals (CrIs).

By default, the Bayesian estimates are based on an MCMC sample size of 10,000 after a 2,500-iteration burn-in. It is important to verify that the MCMC simulation has converged; otherwise, the estimates cannot be trusted. We examine convergence diagnostics later in *Convergence diagnostics*.

An informative prior

Default priors are not provided for user-defined DSGE model parameters. Reasonable priors are model dependent and reflect the role the parameter plays in the model. Share parameters are bounded between 0 and 1; autoregressive parameters are bounded between -1 and 1; some parameters must

be larger than a critical value for model stability. Priors reflect these logical constraints. Additionally, priors reflect information on the probable location of a parameter within that range. Such information can be gleaned through prior studies or through different sources of information, such as microdata or economic theory. Thus priors must be specified for user-defined parameters, and informative priors can affect estimation results.

For standard deviations of shocks, an inverse-gamma prior with the shape and scale parameters of 0.01 is used. You can change the parameters of this prior by specifying the `igammaprior()` option, or you can specify a different prior distribution in the `prior()` option.

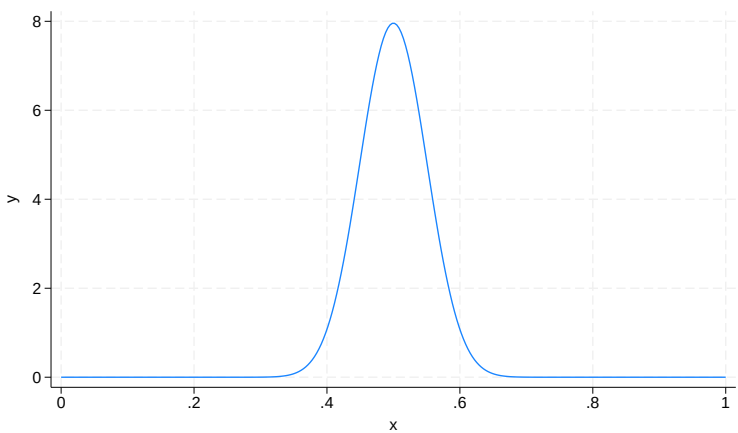
Priors provide several useful functions in DSGE models. Many parameters in DSGE models can be interpreted as share parameters: the capital share, the depreciation rate, the discount rate, etc. Such parameters are naturally bounded by the interval (0,1), and a uniform prior on (0,1) guarantees that the estimated parameter value lies within the economically sensible range. The beta distribution is frequently useful for share parameters, in which greater weight is to be given to a particular region of the (0,1) interval.

In addition, many parameters in DSGE models have microeconomic interpretation. As such, microeconomic information can be brought to bear on the estimation by using informative priors.

In Stata, priors are specified using their statistical parameters as arguments. A uniform prior on (a, b) is specified with `uniform(`*a*`,` *b*`)`. A normal prior with mean μ and variance σ^2 is specified with `normal(`*mu*`,` *sigma2*`)`. A beta distribution with shape parameters α and β is specified with `beta(`*alpha*`,` *beta*`)`. Uniform priors are useful for specifying an uninformative prior on a specified range. Normal priors are useful for parameters that can take on any real value. Beta priors are useful for parameters that are naturally bounded between 0 and 1, a case that is especially salient for the parameters of DSGE models. For a full list of priors, see [BAYES] **bayes**.

We next use an informative prior for `{rho}` on (0,1), centered at 0.5. An appropriate distribution for this task is the beta distribution, and a fairly tight prior is obtained with the shape parameters (50, 50). A graph of the prior is

```
. twoway function y = betaden(50, 50, x)
```



Estimation with the `beta(50,50)` prior is accomplished using the following command:

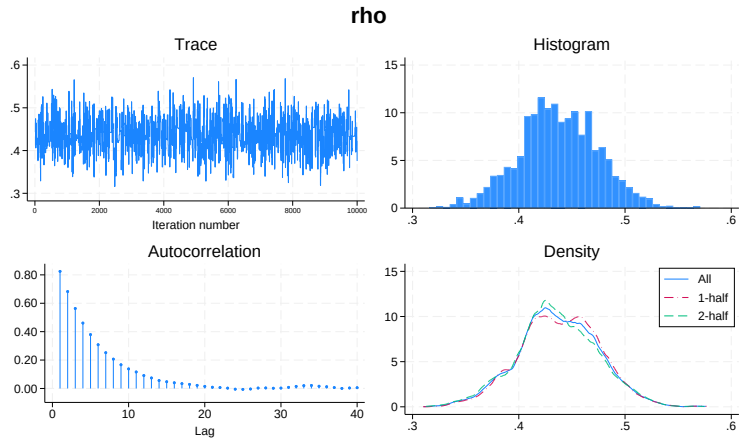
```
. bayes, rseed(17) prior({rho}, beta(50,50)): dsge (y=z) (F.z={rho}*z, state)
note: initial parameter vector set to means of priors.
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  y ~ dsgell({rho},{sd(e.z)})
Priors:
      {rho} ~ beta(50,50)
  {sd(e.z)} ~ igamma(.01,.01)
------------------------------------------------------------------------------

Bayesian linear DSGE model                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
Sample: 1955q1 thru 2015q4                      Number of obs    =        244
                                                Acceptance rate  =      .1459
                                                Efficiency:  min =     .06455
                                                             avg =      .0822
Log marginal-likelihood =  -648.3465                         max =     .09986
------------------------------------------------------------------------------
                                                              Equal-tailed
             |      Mean   Std. dev.     MCSE     Median  [95% cred. interval]
-------------+----------------------------------------------------------------
         rho |  .437131   .0373697   .001183   .4355395   .3616302   .5088862
      sd(e.z)|  3.35469   .1570423   .006181   3.344853   3.057471   3.684154
------------------------------------------------------------------------------
```

The acceptance rate is 15%, somewhat lower than the example with an uninformative prior. Sampling efficiencies are little changed at around 10%. The informative prior has pulled the point estimate of `{rho}` away from its maximum likelihood estimate of 0.34; the posterior mean is now 0.437. The posterior mean for `{rho}` lies about halfway between the prior mean of 0.5 and the maximum likelihood estimate of 0.34. The posterior distribution reflects a balance between prior information and the data at hand. More informative priors, indicated by a smaller standard deviation around the prior mean, tip the balance toward the prior mean. Larger sample sizes increase the importance of the data at hand, though macroeconomic datasets often face practical limitations on the available sample size.

Convergence diagnostics

Continuing with the previous example, we use `bayesgraph diagnostics` to plot diagnostic charts for the autoregressive parameter `{rho}`. Four plots are provided: the MCMC draws, a histogram of the posterior distribution, the autocorrelation of the draws, and a density plot.

```
. bayesgraph diagnostics {rho}
```



We use these plots to visually assess two aspects of the MCMC chain: convergence and efficiency. There are no single conclusive convergence criteria for the MCMC chain, but some visual diagnostics can be used to explore MCMC convergence. (You can also run multiple chains to compute Gelman–Rubin convergence diagnostics; see *Convergence diagnostics using multiple chains*.) The trace plot should not exhibit any time trend and should have constant mean and variance. These properties can be inspected in the trace plot in the top left. The density of the chain should not vary over the duration of the MCMC sample. Constancy of the distribution can be assessed in the 1-half and 2-half density plots in the bottom right; if these plots differ dramatically, then the chain has not converged. For further discussion of convergence, see *Convergence of MCMC*.

Efficiency and mixing are terms used to describe how quickly the MCMC chain traverses the posterior domain. A chain that mixes well has high efficiency, low autocorrelation, and traverses the posterior quickly. A chain that mixes poorly has low efficiency and traverses the posterior slowly, exhibiting high autocorrelation in the MCMC chain. A chain that is less efficient requires more MCMC iterations to obtain the same information as a more efficient chain. Efficiency can also be assessed numerically; see [BAYES] **bayesstats ess**.

For our autoregressive model, the trace plot shows no time trend and appears to have constant mean and variance. The 1-half and 2-half density plots overlap with each other and with the full chain density. These features are indicators that the chain has converged. The autocorrelation of the chain dies out quickly, indicating that the chain has acceptable efficiency. To verify convergence more formally, we could use the `nchains()` option to run multiple chains and to compute Gelman–Rubin convergence diagnostics.

In [DSGE] **Intro 9a** and [DSGE] **Intro 9b**, we demonstrate Bayesian estimation using some of the models we have already developed in previous sections.

[DSGE] **Intro 9a** demonstrates Bayesian estimation of a linear New Keynesian model.

[DSGE] **Intro 9b** demonstrates Bayesian estimation of a nonlinear stochastic growth model.

Also see

[DSGE] **dsge** — Linear dynamic stochastic general equilibrium models

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

[BAYES] **bayes: dsge** — Bayesian linear dynamic stochastic general equilibrium models

[BAYES] **bayes: dsgenl** — Bayesian nonlinear dynamic stochastic general equilibrium models

[BAYES] **bayes** — Bayesian regression models using the bayes prefix

[BAYES] **Intro** — Introduction to Bayesian analysis

[BAYES] **Glossary**

Title

Intro 9a — Bayesian estimation of a New Keynesian model

Description Remarks and examples Also see

Description

This entry estimates and interprets the parameters of a simple New Keynesian model using Bayesian methods. We also discuss some postestimation features, diagnostic tests, and how to improve the sampling efficiency of parameters.

Remarks and examples

Remarks are presented under the following headings:

The model

Equations (1)–(5) specify a canonical New Keynesian model of inflation p_t, the output gap x_t, and the interest rate r_t. The linearized model is

$$p_t = \beta E_t(p_{t+1}) + \kappa x_t \tag{1}$$
$$x_t = E_t(x_{t+1}) - \{r_t - E_t(p_{t+1}) - g_t\} \tag{2}$$
$$r_t = \frac{1}{\psi} p_t + u_t \tag{3}$$
$$u_{t+1} = \rho_u u_t + \epsilon_{t+1} \tag{4}$$
$$g_{t+1} = \rho_g g_t + \xi_{t+1} \tag{5}$$

Equation (1) specifies inflation as a linear combination of expected future inflation and the output gap. Equation (2) specifies the output gap as a linear combination of the expected future output gap, the real interest rate, and a state variable g_t. Equation (3) specifies the interest rate as a linear combination of inflation and a state variable u_t. The state variables are modeled as first-order autoregressive processes. The state variable u_t is the deviation of r_t from its equilibrium value of $(1/\psi)p_t$. The state variable g_t is also the deviation of x_t from its equilibrium value.

Three of the parameters have a structural interpretation. The parameter κ is known as the slope of the Phillips curve and is predicted to be positive. The parameter β is the discount factor that represents the degree to which agents discount the future relative to the current period. The parameter $1/\psi$ measures the degree to which interest rates react to movements in inflation.

The choice of notation is intentional and indicates one way to use priors and parameter transformations. For model stability, the coefficient $1/\psi$ must be greater than one. For theoretical reasons, the region (1.5, 2) is of particular interest. As such, the parameter ψ must lie between 0 and 1. By writing the model in terms of ψ, we use a beta distribution to restrict its range and to put larger weight on the region that is of interest. For example, a beta distribution for ψ that is centered at 0.67 maps on to a prior for $1/\psi$ that places much of its mass around 1.5, and this would be a good mix of logical and theoretical restrictions.

Parameter estimation

We specify priors for the model parameters. The discount rate β must lie between 0 and 1, with common values in the range (0.90, 0.99). The price-adjustment parameter κ is usually thought to be small and positive. The autocorrelation parameters must lie in $(-1, 1)$ but are typically believed to be positive and closer to 1 than 0. The parameter $1/\psi$ must be greater than 1 for model stability; hence, ψ must be between 0 and 1, making the beta distribution a good candidate. The parameters of the beta prior for ψ place its center of mass on 1.5, a value commonly found in the theoretical literature.

Priors were chosen to match the above theoretical considerations. For the discount rate β, a beta distribution with shape parameters (95, 5) is used. These shape parameters place the prior mean at 0.95 and place most of the prior mass in the region between 0.9 and 1. For the price-adjustment parameter κ, a beta distribution with shape parameters (30, 70) is used. These shape parameters place the prior mean at 0.3 and place most of the prior mass in the region between 0.2 and 0.4. For the Taylor rule parameter ψ, a beta distribution with shape parameters (0.67, 0.33) is used. These shape parameters place the prior mean at 0.67, so that its inverse is 1.5, a common value for this parameter in the literature. For the autoregressive parameters, a beta distribution with shape parameters (75, 25) is used. This places the prior mean of each state variable's autoregressive parameter at 0.8, reflecting a prior belief that the state variables show a fairly high degree of persistence.

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

. bayes, prior({beta},  beta(95,  5))
>        prior({kappa}, beta(30, 70))
>        prior({psi},   beta(67, 33))
>        prior({rhou},  beta(75, 20))
>        prior({rhog},  beta(75, 20))
>        rseed(17) :
>        dsge (p   = {beta}*F.p + {kappa}*x)
>             (x   = F.x - (r - F.p - g)   , unobserved)
>             (r   = 1/{psi}*p + u)
>             (F.u = {rhou}*u, state)
>             (F.g = {rhog}*g, state)
note: initial parameter vector set to means of priors.
Burn-in ...
Simulation ...

Model summary
------------------------------------------------------------------------------
Likelihood:
  p r ~ dsgell({beta},{kappa},{psi},{rhou},{rhog},{sd(e.u)},{sd(e.g)})

Priors:
             {beta} ~ beta(95,5)
            {kappa} ~ beta(30,70)
              {psi} ~ beta(67,33)
        {rhou rhog} ~ beta(75,20)
  {sd(e.u) sd(e.g)} ~ igamma(.01,.01)
------------------------------------------------------------------------------
```

```
Bayesian linear DSGE model                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
Sample: 1955q1 thru 2015q4                      Number of obs    =        244
                                                Acceptance rate  =      .2209
                                                Efficiency:  min =      .01093
                                                             avg =      .02799
Log marginal-likelihood = -796.75515                         max =      .05246
```

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred. interval]	
beta	.9330586	.0273878	.002619	.9354095	.8728009	.975816
kappa	.150112	.0247365	.001796	.1485866	.1038851	.203525
psi	.5905049	.0405852	.002398	.5895752	.5146669	.6752993
rhou	.6275518	.0256116	.001511	.6283089	.5764364	.6770674
rhog	.9061189	.0131007	.001059	.9069891	.8800901	.9294626
sd(e.u)	2.114667	.1380783	.006028	2.104597	1.869307	2.400718
sd(e.g)	.5579649	.0578754	.002862	.5574834	.4491618	.6778793

Header output repeats the prior specification and reminds us that we are fitting a DSGE model. The MCMC acceptance rate is 0.2209, with efficiencies ranging from 1% to 5%. An acceptance rate of 20% to 25% is typical for these models. Acceptance rates that are too low indicate that a large portion of the proposed MCMC iterations were rejected, so that regions of high posterior probability were not sufficiently explored. Acceptance rates that are too high indicate that the MCMC iterations stayed in a relatively small area of high probability and did not sufficiently explore the parameter region. Efficiency is linked to the autocorrelation of the MCMC draws, with higher efficiency indicating lower autocorrelation.

Turning to parameter estimates, we see the posterior mean for {beta} is 0.93, close to the prior mean of 0.95. The posterior mean for {kappa} is 0.15, about halfway between the prior mean of 0.3 and the maximum likelihood estimate of 0.08 found in [DSGE] **Intro 3a**. The autoregressive parameters for the state variables are positive, with the state variable g showing autocorrelation {rhog} of 0.91 and the state variable u showing autocorrelation {rhou} of 0.63.

Posterior diagnostics and plots

We begin by examining the efficiency of the MCMC draws.

```
. bayesstats ess
Efficiency summaries    MCMC sample size =    10,000
                        Efficiency:  min =    .01093
                                     avg =    .02799
                                     max =    .05246
```

	ESS	Corr. time	Efficiency
beta	109.35	91.45	0.0109
kappa	189.60	52.74	0.0190
psi	286.40	34.92	0.0286
rhou	287.47	34.79	0.0287
rhog	153.01	65.36	0.0153
sd(e.u)	524.63	19.06	0.0525
sd(e.g)	408.84	24.46	0.0409

Some parameters have sampling efficiencies around 1%, indicating poor mixing. Low efficiency implies that it takes longer for the MCMC chain to explore the posterior distribution. We will come back to the issue of improving sampling efficiency in *Improving sampling efficiency* below.

Sometimes, the object of interest is a function of parameters rather than the parameter itself. Such a situation occurs in this model with the Taylor rule inflation-adjustment parameter $1/\psi$. The output is reported in terms of ψ by default, but what we really want to look at is $1/\psi$. We can analyze the posterior distribution of $1/\psi$ using the postestimation commands available after the `bayes` prefix. The `bayesstats summary` command produces posterior summary statistics for functions of model parameters. We investigate the parameter $1/\psi$ in some detail now.

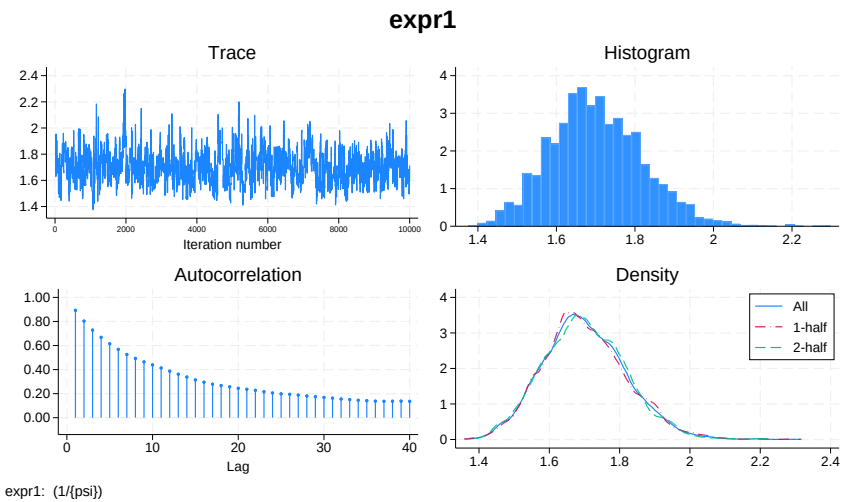
```
. bayesstats summary (1/{psi})
Posterior summary statistics                     MCMC sample size =     10,000
       expr1 : 1/{psi}
```

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred. interval]	
expr1	1.701543	.1181073	.006993	1.696136	1.480825	1.943004

The posterior mean is 1.7, somewhat higher than the prior mean of 1.5.

Next, we investigate the behavior of the MCMC chain using `bayesgraph`. (Actually, this is something that we should have done before obtaining summary statistics to verify that the corresponding MCMC chain converged.) To view the MCMC chain for $1/\psi$, `bayesgraph diagnostics` can be specified with the desired expression directly, bound in parentheses.

```
. bayesgraph diagnostics (1/{psi})
```

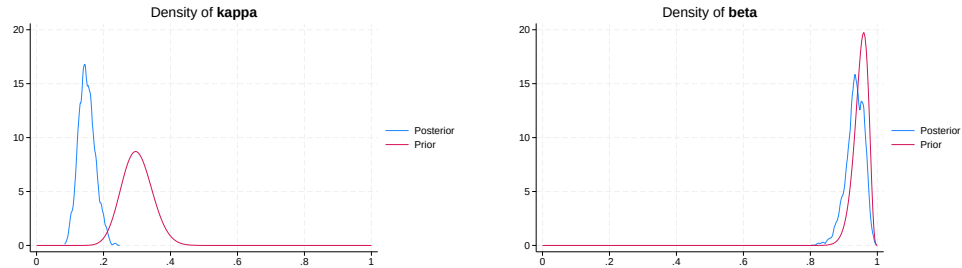


The trace plot shows reasonably good mixing, and the autocorrelations decay at a moderate pace. The density plot shows that the first- and second-half densities are similar to the density of the full MCMC sample. Densities that differ substantially in their first and second half can indicate nonconvergence.

Next, we compare the prior distribution and the posterior distributions for some model parameters directly. The `bayesgraph kdensity` command plots the kernel density of the posterior draws. The `addplot()` option can be used to add the prior to the plot. We plot the prior and posterior of the price-adjustment parameter `{kappa}` and the discount rate `{beta}`.

```
. bayesgraph kdensity {kappa},
> addplot(function Prior = betaden(30,70, x),
> legend(on label(1 "Posterior")))
```

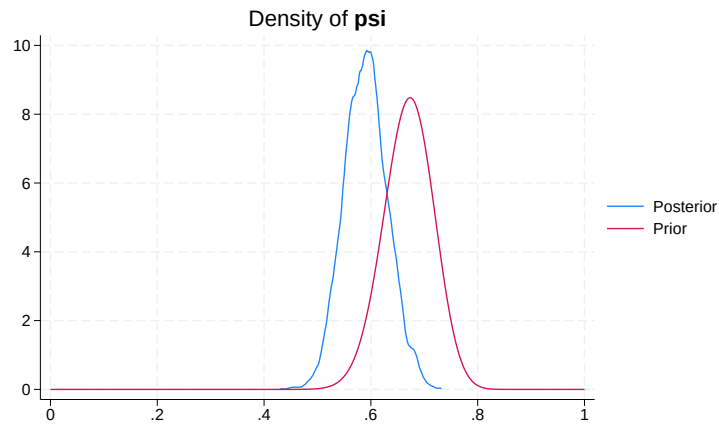
```
. bayesgraph kdensity {beta},
> addplot(function Prior = betaden(95, 5, x),
> legend(on label(1 "Posterior")))
```



The price-adjustment parameter `{kappa}` is now centered on 0.15, rather than the prior mean of 0.3. By contrast, the discount rate parameter `{beta}` has shown little updating; the data are uninformative on this dimension, and the posterior overlaps with the prior.

Next, we turn to the Taylor rule parameter ψ.

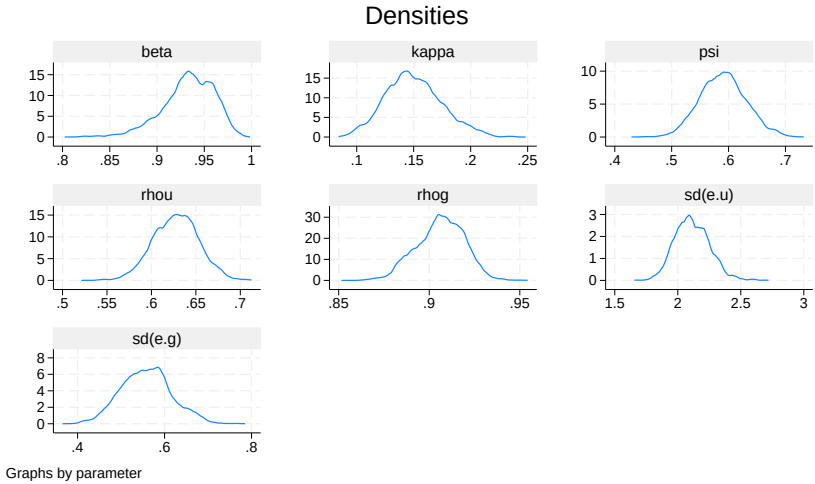
```
. bayesgraph kdensity {psi},
> addplot(function Prior = betaden(67, 33, x),
> legend(on label(1 "Posterior"))
```



The posterior density for ψ places more weight on smaller values compared with the prior, implying a posterior density for $1/\psi$ that places more weight on larger values than the prior.

Finally, `bayesgraph kdensity _all, byparm` provides posterior kernel densities for all model parameters.

```
. bayesgraph kdensity _all, byparm
```



Such a graph can be useful in summarizing the posterior distribution of all parameters. Roughness observed in these densities can be smoothed by running a longer MCMC chain.

Improving sampling efficiency

The `bayesstats ess` output indicates lower sampling efficiency for several model parameters. Blocking of parameters can improve sampling efficiency. Parameters that are to be sampled independently are specified using the `block()` option of `bayes`. Efficiency can also be enhanced by running a longer MCMC chain, accomplished with the `mcmcsize()` option of `bayes`. A longer burn-in period can help with convergence. To demonstrate these procedures, we block some of the parameters and increase the length of the MCMC chain.

```
. bayes, prior({beta},  beta(95,  5))
>        prior({kappa}, beta(30, 70))
>        prior({psi},   beta(67, 33))
>        prior({rhou},  beta(70, 20))
>        prior({rhog},  beta(70, 20))
>        rseed(17)
>        block({kappa}) block({rhou}) block({sd(e.u)})
>        mcmcsize(20000) dots :
>        dsge (p   = {beta}*F.p + {kappa}*x)
>             (x   = F.x - (r - F.p - g)   , unobserved)
>             (r   = 1/{psi}*p + u)
>             (F.u = {rhou}*u, state)
>             (F.g = {rhog}*g, state)
note: initial parameter vector set to means of priors.
Burn-in 2500 aaaaaaaaa1000aaaaaaaaa2000aaaaa done
Simulation 20000 .........1000.........2000.........3000.........4000.........5
> 000.........6000.........7000.........8000.........9000.........10000........
> .11000.........12000.........13000.........14000.........15000.........16000.
> ........17000.........18000.........19000.........20000 done

Model summary
------------------------------------------------------------------------------
Likelihood:
  p r ~ dsgell({beta},{kappa},{psi},{rhou},{rhog},{sd(e.u)},{sd(e.g)})

Priors:
             {beta} ~ beta(95,5)
            {kappa} ~ beta(30,70)
              {psi} ~ beta(67,33)
        {rhou rhog} ~ beta(70,20)
  {sd(e.u) sd(e.g)} ~ igamma(.01,.01)
------------------------------------------------------------------------------

Bayesian linear DSGE model                      MCMC iterations  =     22,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     20,000
Sample: 1955q1 thru 2015q4                      Number of obs    =        244
                                                Acceptance rate  =      .4085
                                                Efficiency:  min =     .01979
                                                             avg =     .03764
Log marginal-likelihood = -796.02145                         max =     .06486

------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. dev.     MCSE     Median  [95% cred. interval]
-------------+----------------------------------------------------------------
        beta |  .9369982   .0266526   .001015   .9401569   .8745056   .9795873
       kappa |  .1543932   .0267139   .001229   .1525291   .1081603   .2125788
         psi |  .5912198   .0415403   .001646   .5908567   .5109676   .6738902
        rhou |  .6201272   .0272482   .001369    .620959   .5658674   .6728684
        rhog |   .905648   .0145937   .000405   .9062614   .8759466   .9333209
      sd(e.u) |  2.11736   .1461866    .00511   2.104432    1.85604   2.439449
      sd(e.g) |  .5544951   .0588497     .0019   .5528083   .4448559   .6758059
------------------------------------------------------------------------------
```

The above command placed three parameters into their own blocks: `{kappa}`, `{rhog}`, and `{sd(e.g)}`. (In this example, we specified three separate `block()` options, but we could have used the shortcut `block({kappa rhog sd(e.g)}, split)`). Blocking improves efficiency at the cost of longer run time. We see that efficiency has improved overall, with the minimum rising to 2% from 1% compared with the estimation without blocking.

```
. bayesstats ess
Efficiency summaries     MCMC sample size =    20,000
                         Efficiency:  min =    .01979
                                      avg =    .03764
                                      max =    .06486
```

	ESS	Corr. time	Efficiency
beta	689.22	29.02	0.0345
kappa	472.47	42.33	0.0236
psi	637.19	31.39	0.0319
rhou	395.87	50.52	0.0198
rhog	1297.26	15.42	0.0649
sd(e.u)	818.49	24.44	0.0409
sd(e.g)	959.44	20.85	0.0480

Effective sample sizes have improved. In particular, the effective sample size for the price-adjustment parameter {kappa} has more than doubled from 189.60 to 472.47.

Impulse responses

To compute Bayesian impulse–response functions, use bayesirf create after estimation. We also need to save off our MCMC dataset. To do this, we replay the bayes command with the saving() option.

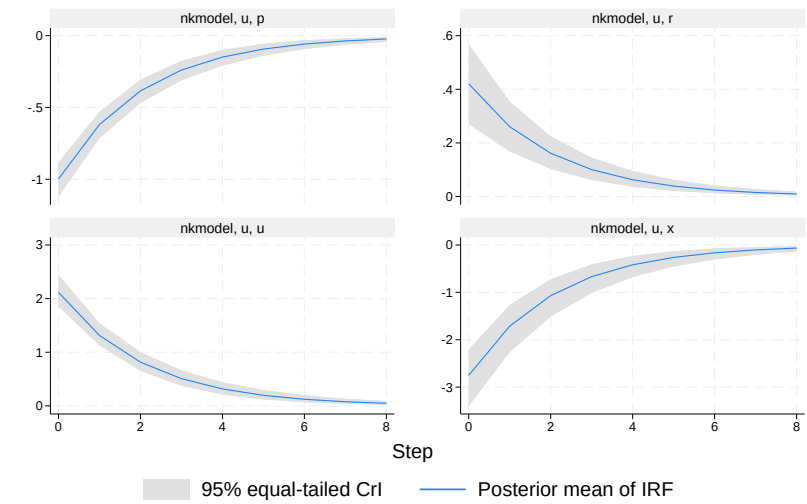
```
. bayes, saving(bdsge_nksim, replace)
```

With the MCMC results saved, we construct impulse–response functions.

```
. bayesirf set bayesirf.irf
. bayesirf create nkmodel
```

Results can be tabulated or graphed with bayesirf table and bayesirf graph, respectively.

```
. bayesirf graph irf, impulse(g) response(x p r g) byopts(yrescale)
```



Graphs by irfname, impulse variable, and response variable

Plotted are the responses of the output gap `x`, inflation `p`, interest rate `r`, and the state variable `u` to a shock to `u`. The steps are in the units of time that were used for estimation, so in this case, one step is one quarter. Eight steps are computed and plotted by default. A rise in the monetary shock causes inflation and interest rates to fall. The output gap becomes negative, indicating that this is a contractionary shock. All three endogenous variables return smoothly to their steady-state values over time, as the effect of the shock dissipates.

Along with point estimates, 95% equal-tailed credible intervals are shown. The point estimates are comparable with those seen in [DSGE] **Intro 1**. But the credible intervals are somewhat narrower than the confidence intervals because of fairly tight priors used.

Also see

[DSGE] **dsge** — Linear dynamic stochastic general equilibrium models

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

[DSGE] **Intro 3a** — New Keynesian model

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

[BAYES] **bayes: dsge** — Bayesian linear dynamic stochastic general equilibrium models

[BAYES] **bayes: dsge postestimation** — Postestimation tools for bayes: dsge and bayes: dsgenl

[BAYES] **bayes** — Bayesian regression models using the bayes prefix

Title

Intro 9b — Bayesian estimation of stochastic growth model

Description Remarks and examples Reference Also see

Description

This introduction estimates and interprets the parameters of a simple stochastic growth model using Bayesian methods.

Remarks and examples

Remarks are presented under the following headings:

The model

The model contains equations that jointly determine output Y_t, the interest rate R_t, consumption C_t, capital K_t, and productivity Z_t. The model contains four parameters: α, β, δ, and ρ. This model is a variant on the model used in Schmitt-Grohé and Uribe (2004):

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

$$Y_t = Z_t K_t^{\alpha} \tag{2}$$

$$R_t = \alpha Z_t K_t^{\alpha-1} \tag{3}$$

$$K_{t+1} = Y_t - C_t + (1-\delta)K_t \tag{4}$$

$$\ln(Z_{t+1}) = \rho \ln(Z_t) + e_{t+1} \tag{5}$$

Equation (1) defines a relationship between consumption growth C_{t+1}/C_t and the real interest rate R_{t+1}. Equation (2) is a production function for output Y_t as a function of productivity Z_t and capital K_t. Equation (3) is a model for the interest rate. Equation (4) is the equation for capital accumulation; capital in the next period is equal to underappreciated capital this period $(1-\delta)K_t$ plus unconsumed output $Y_t - C_t$. Equation (5) is a law of motion for productivity Z_t. The parameter β is a discount factor in the consumption equation, the parameter α is a production parameter in the output equation, the parameter δ is a depreciation parameter in the capital equation, and the parameter ρ is a persistence parameter in the productivity equation.

The state variables are the current-period capital stock and the level of productivity, (K_t, Z_t). The control variables are consumption, the interest rate, and output, (C_t, R_t, Y_t). We estimate the parameters of the linearized version of the model using Bayesian methods.

Parameter estimation

We specify priors for the model parameters. The discount rate β must lie between 0 and 1, with common values in the range (0.95, 0.99). The capital share α must lie between 0 and 1 and is usually estimated to be about one-third. The depreciation rate δ must lie between 0 and 1, and a common value for it is 0.025 or 0.10. The autocorrelation parameter ρ must be less than 1 in absolute value and is usually thought to be positive and close to 1. Our prior choices for all parameters are driven by these theoretical considerations. As all four parameters are plausibly restricted to the unit interval, a beta distribution is chosen for all four priors.

The parameters of the beta distribution were chosen to put the weight of prior mass on theoretically appropriate values. For the discount factor `{beta}`, this is the range (0.90, 0.99). For the depreciation parameter `{delta}`, this is the range (0.03, 0.05). For the capital share `{alpha}`, this is the range (0.3, 0.4). For the autoregressive parameter, this is the range (0.6, 0.99). The prior means for each parameter are as follows: `{beta}` prior mean is 0.95; `{delta}` prior mean is 0.044; `{alpha}` prior mean is 0.38; and `{rhoz}` prior mean is 0.8.

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

. bayes, prior({beta},  beta( 95,   5))
>        prior({delta}, beta( 40, 860))
>        prior({alpha}, beta(360, 590))
>        prior({rhoz},  beta( 80,  20)) dots rseed(17) :
>        dsgenl (1 = {beta}*(c/F.c)*(1 + F.r - {delta}))
>               (y = z*k^({alpha}))
>               (r = {alpha}*y/k)
>               (F.k = y - c + (1-{delta})*k)
>               (ln(F.z) = {rhoz}*ln(z)),
>               exostate(z) endostate(k) observed(y) unobserved(c r)
note: initial parameter vector set to means of priors.
Burn-in 2500 aaaaaaaaa1000aaaaaaaaa2000..... done
Simulation 10000 .........1000.........2000.........3000.........4000.........
> 5000.........6000.........7000.........8000.........9000.........10000 done

Model summary
------------------------------------------------------------------------------

Likelihood:
  y ~ dsgell({beta},{delta},{alpha},{rhoz},{sd(e.z)})

Priors:
     {beta} ~ beta(95,5)
    {delta} ~ beta(40,860)
    {alpha} ~ beta(360,590)
     {rhoz} ~ beta(80,20)
  {sd(e.z)} ~ igamma(.01,.01)
------------------------------------------------------------------------------
```

```
Bayesian first-order DSGE model                 MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
Sample: 1955q1 thru 2015q4                      Number of obs    =        244
                                                Acceptance rate  =      .2479
                                                Efficiency:  min =      .0228
                                                             avg =        .05
Log marginal-likelihood = -670.33605                         max =      .05948
```

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred. interval]	
beta	.9696788	.0138534	.000918	.9714437	.9379939	.9904246
delta	.0412205	.0063456	.000272	.0408097	.029939	.0545771
alpha	.3790517	.0154948	.000645	.3792245	.3486508	.4096519
rhoz	.6178456	.0442668	.001815	.618141	.5340957	.7060575
sd(e.z)	3.54923	.1751019	.007431	3.544049	3.238064	3.931496

The model summary reports the prior and likelihood specifications, including the default inverse-gamma prior for the standard deviation of the shock.

The output header reports the burn-in length and MCMC sample size, as well as information about the efficiency of the Metropolis–Hastings sampler. The overall acceptance rate is 0.25, with sampling efficiencies between 0.023 and 0.059.

The posterior mean for {beta} is 0.97, close to the prior mean of 0.95. The posterior mean for {delta} is 0.041, close to its prior mean of 0.044. The posterior mean for {alpha} is 0.38, identical to its prior mean. The posterior mean for {rhoz} is 0.62, substantially different from its prior mean of 0.80. Overall, many of the parameters show little updating, indicating that the likelihood is uninformative along several dimensions of the model's parameter space. The posterior results for {beta}, {delta}, and {alpha} are mainly driven by the prior.

Posterior diagnostics and plots

We begin by investigating effective sample sizes for each parameter.

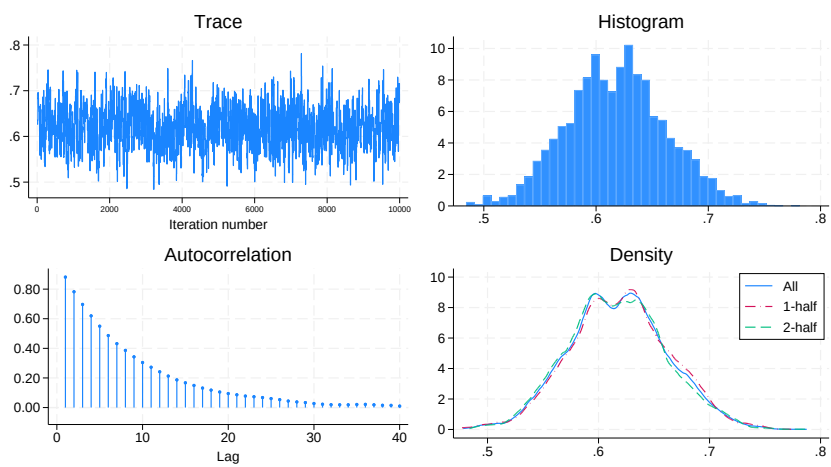
```
. bayesstats ess
Efficiency summaries    MCMC sample size =    10,000
                        Efficiency:  min =     .0228
                                     avg =       .05
                                     max =    .05948
```

	ESS	Corr. time	Efficiency
beta	227.97	43.87	0.0228
delta	545.58	18.33	0.0546
alpha	576.58	17.34	0.0577
rhoz	594.79	16.81	0.0595
sd(e.z)	555.25	18.01	0.0555

The effective sample size for the discount factor {beta} is somewhat low relative to the other parameters, which indicates that blocking may improve sampling efficiency.

Because {rhoz} was the only internal parameter to receive substantial updating, we look at its full set of posterior diagnostic plots.

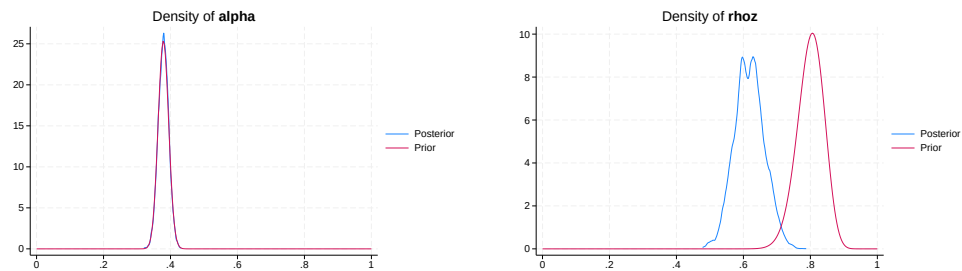
```
. bayesgraph diagnostics {rhoz}
```



Autocorrelations tail off at a moderate pace, the trace plot shows reasonable mixing, and the density plot shows that the first- and second-half densities do not substantially differ from the full-sample density.

Next, we generate prior–posterior plots for two parameters, the capital share and the autoregressive parameter.

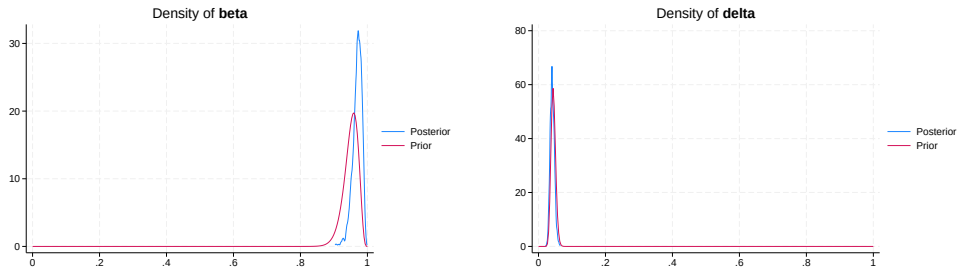
```
. bayesgraph kdensity {alpha},
> addplot(function Prior = betaden(360, 590, x),
> legend(on label(1 "Posterior")))
. bayesgraph kdensity {rhoz},
> addplot(function Prior = betaden(80, 20, x),
> legend(on label(1 "Posterior")))
```



The posterior density of {alpha} does not differ from its prior density. This situation indicates a flat likelihood along the {alpha} dimension. By contrast, the posterior density of {rhoz} does differ from its prior density. The posterior mean has fallen to 0.6 from 0.8.

Prior–posterior plots for the discount rate {beta} and the depreciation parameter {delta} are

```
. bayesgraph kdensity {beta},
> addplot(function Prior = betaden(95, 5, x),
> legend(on label(1 "Posterior")))
. bayesgraph kdensity {delta},
> addplot(function Prior = betaden(40, 860, x),
> legend(on label(1 "Posterior")))
```



The posterior distribution of {beta} lies close to its prior. The posterior distribution of {delta} lies on top of its prior, indicating that the data provide little information along this dimension of the model.

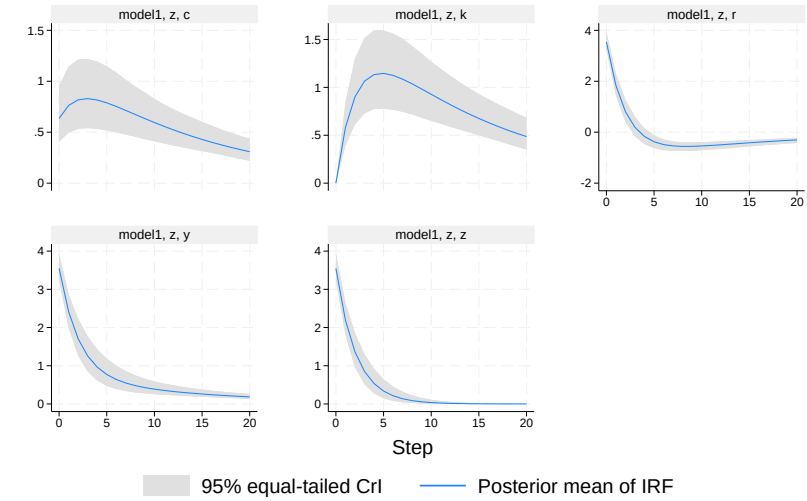
Impulse responses

Next, we explore the response of model variables to a shock to the state variable z. We begin by saving off our MCMC results in a dataset.

```
. bayes, saving(bayes_dsgenl_sim, replace)
```

Next, we set up the impulse–response function file and impulse–response functions themselves. We are using the [BAYES] **bayesirf** command. We specify step(20) to plot the first 20 periods after the shock. Because the unit of time in this model is one quarter, 20 periods correspond to 5 years.

```
. bayesirf set rbcmodel
. bayesirf create model1, step(20)
. bayesirf graph irf, impulse(z) byopts(yrescale)
```



Graphs by irfname, impulse variable, and response variable

Each panel displays the response of one model variable to the impulse. Each step is one quarter, so that four steps are one year after the shock. In the top-left panel, consumption `c` rises and follows a mostly flat trajectory for the first eight periods after the shock and then falls to return to its steady state. In the top-middle panel, the capital stock `k` does not move in the first period but rises afterward in a hump-shaped pattern. In the top-right panel, the interest rate `r` rises on impact, remains elevated for the first four periods, and then dips below its steady-state value in the fifth period; it returns to its steady state from below. In the bottom-left panel, output `y` rises on impact and then declines monotonically back to its steady state. The bottom-right panel shows the evolution of the state variable `z` itself; it rises on a shock and then falls monotonically back to its steady state.

Reference

Schmitt-Grohé, S., and M. Uribe. 2004. Solving dynamic general equilibrium models using a second-order approximation to the policy function. *Journal of Economic Dynamics and Control* 28: 755–775. https://doi.org/10.1016/S0165-1889(03)00043-5.

Also see

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

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

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

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

[BAYES] **bayes: dsgenl** — Bayesian nonlinear dynamic stochastic general equilibrium models

[BAYES] **bayes: dsge postestimation** — Postestimation tools for bayes: dsge and bayes: dsgenl

[BAYES] **bayes** — Bayesian regression models using the bayes prefix

Title

dsge — Linear dynamic stochastic general equilibrium models

Description

`dsge` fits linear dynamic stochastic general equilibrium (DSGE) models to multiple time series. DSGE models are systems of equations that are motivated by economic theory. In these systems, expectations of future values of variables can affect the current values. The parameters of these models are often directly interpretable in terms of economic theory.

Menu

Statistics > Multivariate time series > Dynamic stochastic general equilibrium (DSGE) models > Linear DSGE models

Syntax

`dsge` *eqlist* [*if*] [*in*] [, *options*]

options	Description
Main	
`constraints(`*constraints*`)`	apply specified linear constraints
`noidencheck`	do not check for parameter identification
`solve`	return model solution at initial values
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim` or `robust`
Reporting	
`level(#)`	set confidence level; default is `level(95)`
`nocnsreport`	do not display constraints
display_options	control columns and column formats and line width
Maximization	
maximize_options	control the maximization process
Advanced	
`nodemean`	do not demean data prior to estimation
`post`	force posting of estimation results in the event of errors caused by lack of identification or stability
`idtolerance(#)`	set tolerance used for identification check; seldom used
`lintolerance(#)`	set tolerance used for linearity check; seldom used
`coeflegend`	display legend instead of statistics

You must `tsset` your data before using `dsge`; see [TS] **tsset**.
`bayes` and `collect` are allowed; see [U] **11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: dsge**.
`coeflegend` does not appear in the dialog box.
See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

Below we present the full specification of *eqlist*. You may prefer to start with the syntax discussion in [DSGE] **Intro 2**.

eqlist is
eq
eq eqlist

eq is
cntrl_eq
state_eq

cntrl_eq contains
(*ocntrl_var* = *termlist*)
(*ucntrl_var* = *termlist*, unobserved)
(*parm_exp* * *ocntrl_var* = *termlist*)
(*parm_exp* * *ucntrl_var* = *termlist*, unobserved)

state_eq contains
(F.*state_var* = , state)
(F.*state_var* = *termlist*, state [noshock])
(*parm_exp* * F.*state_var* = *termlist*, state [noshock])

ocntrl_var is Stata variable to be treated as an observed control in the system
ucntrl_var is name to be treated as an unobserved control in the system
state_var is name to be treated as an unobserved state in the system

If there happens to be a Stata variable with the same name as *ucntrl_var* or *state_var*, the variable is ignored and plays no role in the estimation.

termlist is
term
term + *termlist*
term - *termlist*

term is
rhs_var
parm_exp * *rhs_var*
parm_exp * (*termlist*)

rhs_var is
ocntrl_var
ucntrl_var
state_var
F.*ocntrl_var*
F.*ucntrl_var*

parm_exp is	scalar substitutable expression; *parm_spec* elements are allowed and expected *rhs_var* are not allowed
parm_spec is	
	{*parm_name*}
	{*parm_name*=*initial_val*}
parm_name is	name used to identify model parameter
initial_val is	numeric literal; specifies an initial value

Options

Main

`constraints(`*constraints*`)`; see [R] **Estimation options**.

`noidencheck` skips the check that the parameters are identified at the initial values. Models that are not structurally identified can still converge, thereby producing meaningless results that only appear to have meaning; thus care should be taken in specifying this option. See [DSGE] **Intro 6** for details.

`solve` puts the model into state-space form at the initial parameter values. No standard errors are produced.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory (`oim`) and that are robust to some kinds of misspecification (`robust`); see [R] ***vce_option***.

Reporting

`level(#)`, `nocnsreport`; see [R] **Estimation options**.

display_options: `noci`, `nopvalues`, `cformat(%`*fmt*`)`, `pformat(%`*fmt*`)`, `sformat(%`*fmt*`)`, and `nolstretch`; see [R] **Estimation options**.

Maximization

maximize_options: `difficult`, `technique(`*algorithm_spec*`)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, and `from(`*init_specs*`)`; see [R] **Maximize**.

Advanced

`nodemean` prevents `dsge` from removing the mean of the observed control variables prior to estimation.

`post` causes `dsge` to post the parameter vector into `e()`, even in the event of errors that arise from checking stability conditions or identification.

`idtolerance(#)` specifies the tolerance used in the identification diagnostic. The default is `idtolerance(1e-6)`.

`lintolerance(#)` specifies the tolerance used in the linearity diagnostic. The default is `lintolerance(1e-12)`.

The following option is available with dsge but is not shown in the dialog box:

coeflegend; see [R] **Estimation options**.

Remarks

For an introduction to what dsge can do and how to use it, see [DSGE] **Intro 1**. It is highly recommended that you read the introduction first.

For examples of dsge, see the examples of classic DSGE models in [DSGE] **Intro 3a**, [DSGE] **Intro 3b**, and [DSGE] **Intro 3c**. Additional examples are presented in [DSGE] **Intro 4a**–[DSGE] **Intro 4g**. See [DSGE] **Intro 4** for an overview of these examples.

Stored results

dsge stores the following in e():

Scalars

e(N)	number of observations
e(k)	number of parameters
e(k_eq)	number of equations in e(b)
e(k_dv)	number of dependent variables
e(k_state)	number of state equations
e(k_control)	number of control equations
e(k_shock)	number of shocks
e(k_observed)	number of observed control equations
e(k_stable)	number of stable eigenvalues
e(ll)	log likelihood
e(tmin)	minimum time in sample
e(tmax)	maximum time in sample
e(rank)	rank of VCE
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	dsge
e(cmdline)	command as typed
e(depvar)	unoperated names of dependent variables
e(state)	unoperated names of state variables
e(control)	unoperated names of control variables
e(observed)	unoperated names of observed control variables
e(title)	title in estimation output
e(tvar)	variable denoting time within groups
e(tmins)	formatted minimum time
e(tmaxs)	formatted maximum time
e(tsfmt)	format for the current time variable
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(method)	likelihood method
e(idencheck)	passed, failed, or skipped
e(technique)	maximization technique
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins

Matrices

e(b)	parameter vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(eigenvalues)	generalized eigenvalues
e(gradient)	gradient vector
e(shock_coeff)	estimated shock coefficient matrix
e(transition)	estimated state transition matrix
e(policy)	estimated policy matrix
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

A DSGE model is a system of equations in which the values of the control variables $\mathbf{y}_t$ and of the state variables $\mathbf{x}_t$ depend on their values in previous periods and their one-period-ahead rational expectations; see [DSGE] **Intro 1** for an introduction to control variables and state variables. The structural form of a DSGE model is

$$\mathbf{A}_0\mathbf{y}_t = \mathbf{A}_1 E_t(\mathbf{y}_{t+1}) + \mathbf{A}_2\mathbf{y}_t + \mathbf{A}_3\mathbf{x}_t \tag{1}$$

$$\mathbf{B}_0\mathbf{x}_{t+1} = \mathbf{B}_1 E_t(\mathbf{y}_{t+1}) + \mathbf{B}_2\mathbf{y}_t + \mathbf{B}_3\mathbf{x}_t + \mathbf{C}\boldsymbol{\epsilon}_{t+1} \tag{2}$$

where the entries in $\mathbf{A}_i$ and $\mathbf{B}_i$ are functions of the parameter vector $\boldsymbol{\theta}$ and $\mathbf{C}$ is a selector matrix of zeros and ones that puts shocks in the state equations that include them. The $\mathbf{A}_0$ and $\mathbf{B}_0$ matrices are diagonal, and $\mathbf{A}_2$ has zeros on its diagonal. See [DSGE] **Intro 1** for details about how the structural parameters $\boldsymbol{\theta}$ get mapped into entries in $\mathbf{A}_i$ and $\mathbf{B}_i$ matrices.

dsge estimates the structural parameters $\boldsymbol{\theta}$ by maximum likelihood. The remainder of this entry provides some details about this process.

We can estimate only the parameters of structural models like (1) and (2) that we can solve for the state-space form that expresses the control variables as functions of the state variables and expresses the state variables as first-order autoregressive processes, as in (3) and (4),

$$\mathbf{y}_t = \mathbf{G}(\boldsymbol{\theta})\mathbf{x}_t \tag{3}$$

$$\mathbf{x}_{t+1} = \mathbf{H}(\boldsymbol{\theta})\mathbf{x}_t + \mathbf{M}(\boldsymbol{\theta})\boldsymbol{\epsilon}_{t+1} \tag{4}$$

where $\mathbf{G}$ is known as the policy matrix and $\mathbf{H}$ is known as the state transition matrix. The elements in the policy and state transition matrices are functions of the structural parameters that appear in the matrices $\mathbf{A}$ and $\mathbf{B}$.

dsge implements the solution technique described in Klein (2000), which obtains the state-space form in (3) and (4) from the structural model in (1) and (2).

We can only solve a structural model for its state-space form when i) the structural form has the structure implied by (1) and (2) and when ii) the values of the structural parameters imply that the model neither spirals out of control nor converges to a point, a condition known as saddle-path stability.

Condition (i) seems trivial, but some minor manipulation is frequently required to make it true. Many DSGE models include a problematic term like a shock to a control variable or a lagged state variable. These problematic terms do not fit into the structure required by (1) and (2), but the model is easily rewritten to accommodate these problematic terms; see [DSGE] **Intro 4** for details.

Condition (ii), the saddle-path stability condition, depends on the values of the parameters. We can only solve models for parameter values that imply a saddle-path-stable model; see [DSGE] **Intro 5** for details.

Assuming that the shocks $\boldsymbol{\epsilon}_t$ are mean-zero, independent, and normally distributed implies that (3) and (4) form a linear state-space model in which the coefficients are nonlinear functions of the structural parameters $\boldsymbol{\theta}$. This structure allows `dsge` to estimate the parameters $\boldsymbol{\theta}$ and the standard deviations of the shocks by maximum likelihood.

The Kalman filter is used to form the log-likelihood function. The Kalman filter is a method for recursively obtaining linear, least-squares forecasts conditional on past information. These forecasts are used to construct the log likelihood, assuming normality and stationarity. See *Methods and formulas* in [TS] **sspace** for details of the Kalman filter.

When the shocks are independent and identically distributed, but not normally distributed, the maximum likelihood estimator is consistent for $\boldsymbol{\theta}$, but `vce(robust)` standard errors must be used. (The process of using a maximum likelihood estimator under weaker distributional assumptions and correcting the standard errors is known as a quasimaximum likelihood estimator in the literature.)

One final caveat is that only a subset $\mathbf{y}_{1,t}$ of the controls is treated as observed in some models. This caveat is handled by augmenting the state-space systems in (3) and (4) with the observation equation

$$\mathbf{y}_{1,t} = \mathbf{D}\mathbf{y}_t$$

where $\mathbf{D}$ is a selection matrix that picks out the rows in $\mathbf{G}$ corresponding to elements in the control vector that are observed.

Asymptotic standard errors for postestimation objects such as the entries in the state-space matrices and the impulse responses are obtained using the delta method. See Serfling (1980, sec. 3.3) for a discussion of the delta method.

References

Klein, P. 2000. Using the generalized Schur form to solve a multivariate linear rational expectations model. *Journal of Economic Dynamics and Control* 24: 1405–1423. https://doi.org/10.1016/S0165-1889(99)00045-7.

Schenck, D. 2017. Estimating the parameters of DSGE models. *The Stata Blog: Not Elsewhere Classified.* https://blog.stata.com/2017/07/11/estimating-the-parameters-of-dsge-models/.

——. 2018. Dynamic stochastic general equilibrium models for policy analysis. *The Stata Blog: Not Elsewhere Classified.* https://blog.stata.com/2018/04/23/dynamic-stochastic-general-equilibrium-models-for-policy-analysis/.

Serfling, R. J. 1980. *Approximation Theorems of Mathematical Statistics.* New York: Wiley.

Also see

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

[DSGE] **Intro 2** — Learning the syntax

[BAYES] **bayes: dsge** — Bayesian linear dynamic stochastic general equilibrium models

[TS] **sspace** — State-space models

[TS] **tsset** — Declare data to be time-series data

[TS] **var** — Vector autoregressive models

[U] **20 Estimation and postestimation commands**

Title

dsge postestimation — Postestimation tools for dsge

Postestimation commands

The following postestimation commands are of special interest after `dsge`:

Command	Description
`estat policy`	display policy matrix of estimated model
`estat stable`	assess stability of the system
`estat transition`	display transition matrix of estimated model
`irf`	create and analyze IRFs and FEVDs

The following standard postestimation commands are also available:

Command	Description
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estimates`	cataloging estimation results
`etable`	table of estimation results
`forecast`	dynamic forecasts and simulations
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of coefficients
`lrtest`	likelihood-ratio test
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients
`predict`	one-step-ahead predictions, prediction errors, and other diagnostic measures
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

predict

Description for predict

`predict` creates new variables containing predictions such as expected values. Predictions are available as static one-step-ahead predictions or as dynamic multistep predictions, and you can control when dynamic predictions begin.

Menu for predict

Statistics > Postestimation

Syntax for predict

`predict` [*type*] {*stub*`*`|*newvarlist*} [*if*] [*in*] [`,` *statistic options*]

statistic	Description
Main	
`xb`	linear prediction for observed variables
`states`	linear prediction for latent state variables

options	Description
Options	
`rmse(`*stub*`*`|*newvarlist*`)`	put estimated root mean squared errors of predicted statistics in new variables
`dynamic(`*time_constant*`)`	begin dynamic forecast at specified time
Advanced	
`smethod(`*method*`)`	method for predicting unobserved states

method	Description
`onestep`	predict using past information
`filter`	predict using past and contemporaneous information

Options for predict

Main

`xb`, the default, calculates the linear predictions of the observed variables.

`states` calculates the linear predictions of the latent state variables.

Options

`rmse(`*stub*`*` | *newvarlist*`)` puts the root mean squared errors of the predicted statistics into the specified new variables. The root mean squared errors measure the variances due to the disturbances but do not account for estimation error.

`dynamic(`*time_constant*`)` specifies when `predict` starts producing dynamic forecasts. The specified *time_constant* must be in the scale of the time variable specified in `tsset`, and the *time_constant* must be inside a sample for which observations on the dependent variables are available. For example, `dynamic(tq(2008q4))` causes dynamic predictions to begin in the fourth quarter of 2008, assuming that your time variable is quarterly; see [D] **Datetime**. If the model contains exogenous variables, they must be present for the whole predicted sample.

Advanced

`smethod(`*method*`)` specifies the method for predicting the unobserved states, `smethod(onestep)` and `smethod(filter)`, and causes different amounts of information on the dependent variables to be used in predicting the states at each time period.

`smethod(onestep)`, the default, causes `predict` to estimate the states at each time period using previous information on the dependent variables. The Kalman filter is performed on previous periods, but only the one-step predictions are made for the current period.

`smethod(filter)` causes `predict` to estimate the states at each time period using previous and contemporaneous data by the Kalman filter. The Kalman filter is performed on previous periods and the current period. `smethod(filter)` may be specified only with `states`.

Remarks and examples

For examples of `estat policy`, see [DSGE] **Intro 1**, [DSGE] **Intro 3a**, and [DSGE] **Intro 3c**.

For examples of `estat transition`, see [DSGE] **Intro 1**, [DSGE] **Intro 3a**, and [DSGE] **Intro 3b**.

For an example of `estat stable`, see [DSGE] **Intro 5**.

For examples of `irf` after `dsge`, see [DSGE] **Intro 1**, [DSGE] **Intro 3b**, and [DSGE] **Intro 3c**.

For an example of `forecast` after `dsge`, see [DSGE] **Intro 1**.

For examples of `predict` after `dsge`, see [DSGE] **Intro 3a**.

Methods and formulas

Estimating the unobserved states is the key to predicting the observed variables.

By default and with the `smethod(onestep)` option, `predict` estimates the state in each period by applying the Kalman filter to all previous periods and only making the one-step prediction to the current period.

With the `smethod(filter)` option, `predict` estimates the states in each period by applying the Kalman filter on all previous periods and the current period. The computational difference between `smethod(onestep)` and `smethod(filter)` is that `smethod(filter)` performs the update step on the current period while `smethod(onestep)` does not. The statistical difference between `smethod(onestep)` and `smethod(filter)` is that `smethod(filter)` uses contemporaneous information on the observed variables while `smethod(onestep)` does not.

The observed control variables are predicted by plugging in the estimated states.

Also see

[DSGE] **dsge** — Linear dynamic stochastic general equilibrium models

[DSGE] **estat policy** — Display policy matrix

[DSGE] **estat stable** — Check stability of system

[DSGE] **estat transition** — Display state transition matrix

[TS] **forecast** — Econometric model forecasting

[TS] **irf** — Create and analyze IRFs, dynamic-multiplier functions, and FEVDs

[U] **20 Estimation and postestimation commands**

Title

dsgenl — Nonlinear dynamic stochastic general equilibrium models

Description

`dsgenl` fits nonlinear dynamic stochastic general equilibrium (DSGE) models to multiple time series via perturbation methods. DSGE models are systems of equations that are motivated by economic theory. In these systems, expectations of future values of variables can affect the current values. The parameters of these models are often directly interpretable in terms of economic theory.

Menu

Statistics > Multivariate time series > Dynamic stochastic general equilibrium (DSGE) models > Nonlinear DSGE models

Syntax

```
dsgenl (lexp_1 = rexp_1) [(lexp_2 = rexp_2) ...] [if] [in],
  observed(varlist) exostate(namelist) [options]
```

rexp_j and *lexp_j* are substitutable expressions on the right-hand side and left-hand side of equation *j*. These are Stata expressions that include scalars, variables, and parameters to be estimated. The variables may be state variables, observed control variables, and unobserved control variables. The lead operator, `F.`, can be applied to a variable so that `F.`*varname* refers to the expected value of *varname* in the next time period. No other time-series operators are allowed; see [DSGE] **Intro 4** for information on including additional lags and leads. Parameters to be estimated are enclosed in curly braces; `{beta}` is an example of a parameter. Initial values for parameters are given by including an equal sign and the initial value inside the braces; for example, `{beta=2}` sets the initial value for `beta` to 2. See *Specifying the system of nonlinear equations* in [DSGE] **Intro 2**.

options	Description
Main	
* observed(*varlist*)	list of observed control variables
unobserved(*namelist*)	list of unobserved control variables
* exostate(*namelist*)	list of exogenous state variables
endostate(*namelist*)	list of endogenous state variables
constraints(*constraints*)	apply specified linear constraints
linearapprox	take a linear, rather than log-linear, approximation
noidencheck	do not check for parameter identification
solve	return model solution at initial values
SE/Robust	
vce(*vcetype*)	*vcetype* may be oim or robust
Reporting	
level(#)	set confidence level; default is level(95)
nocnsreport	do not display constraints
display_options	control columns and column formats and line width
Maximization	
maximize_options	control the maximization process
Advanced	
nodemean	do not demean data prior to estimation
post	force posting of estimation results in the event of errors caused by lack of identification or stability
idtolerance(#)	set tolerance used for identification check; seldom used
steadytolerance(#)	set tolerance used for convergence in steady-state calculations; seldom used
steadyinit(*matrix*)	set initial values for steady state; seldom used
coeflegend	display legend instead of statistics

* observed() and exostate() are required.

You must tsset your data before using dsgenl; see [TS] **tsset**.

bayes and collect are allowed; see [U] **11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: dsgenl**.

coeflegend does not appear in the dialog box.

See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Main

observed(*varlist*) specifies which variables that appear in the model equations are observed control variables. observed() is required.

unobserved(*namelist*) specifies which variables that appear in the model equations are unobserved control variables. The names must be valid Stata names but need not exist in your dataset.

exostate(*namelist*) specifies which variables that appear in the model equations are exogenous state variables. Exogenous state variables are subject to shocks. The names must be valid Stata names but need not exist in your dataset. There must be the same number of exogenous state variables as there are observed control variables in your model. exostate() is required.

`endostate(`*namelist*`)` specifies which variables that appear in the model equations are endogenous state variables. Endogenous state variables are not subject to shocks. The names must be valid Stata names but need not exist in your dataset.

`constraints(`*constraints*`)`; see [R] **Estimation options**.

`linearapprox` specifies that a linear approximation be applied to the DSGE model rather than a log-linear approximation. In either case, an approximation is applied. In a log-linear approximation, variables are interpreted as percentage deviations from steady state. In a linear approximation, variables are interpreted as unit deviations from steady state.

`noidencheck` skips the check that the parameters are identified at the initial values. Models that are not structurally identified can still converge, thereby producing meaningless results that only appear to have meaning; thus, care should be taken in specifying this option. See [DSGE] **Intro 6** for details.

`solve` puts the model into state-space form at the initial parameter values. No standard errors are produced.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory (`oim`) and that are robust to some kinds of misspecification (`robust`); see [R] ***vce_option***.

Reporting

`level(#)`, `nocnsreport`; see [R] **Estimation options**.

display_options: `noci`, `nopvalues`, `cformat(%`*fmt*`)`, `pformat(%`*fmt*`)`, `sformat(%`*fmt*`)`, and `nolstretch`; see [R] **Estimation options**.

Maximization

maximize_options: `difficult`, `technique(`*algorithm_spec*`)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, and `from(`*init_specs*`)`; see [R] **Maximize**.

Advanced

`nodemean` prevents `dsgenl` from removing the mean of the observed control variables prior to estimation.

`post` causes `dsgenl` to post the parameter vector into `e()`, even in the event of errors that arise from checking stability conditions or identification.

`idtolerance(#)` specifies the tolerance used in the identification diagnostic. The default is `idtolerance(1e-6)`.

`steadytolerance(#)` specifies the tolerance used in calculations of the model's steady state. The default is `steadytolerance(1e-17)`.

`steadyinit(`*matrix*`)` specifies a vector of initial values for use in finding the steady state.

The following option is available with `dsgenl` but is not shown in the dialog box:

`coeflegend`; see [R] **Estimation options**.

Remarks

For an introduction to what dsgenl can do and how to use it, see [DSGE] **Intro 1**. It is highly recommended that you read the introduction first.

For examples of dsgenl, see the examples of classic DSGE models in [DSGE] **Intro 3d**, [DSGE] **Intro 3e**, and [DSGE] **Intro 3f**.

Stored results

dsgenl stores the following in e():

Scalars

e(N)	number of observations
e(k)	number of parameters
e(k_eq)	number of equations in e(b)
e(k_dv)	number of dependent variables
e(k_state)	number of state equations
e(k_control)	number of control equations
e(k_shock)	number of shocks
e(k_observed)	number of observed control equations
e(k_stable)	number of stable eigenvalues
e(ll)	log likelihood
e(tmin)	minimum time in sample
e(tmax)	maximum time in sample
e(rank)	rank of VCE
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	dsgenl
e(cmdline)	command as typed
e(depvar)	unoperated names of dependent variables
e(state)	unoperated names of state variables
e(shock)	unoperated names of state variables subject to shocks
e(control)	unoperated names of control variables
e(observed)	unoperated names of observed control variables
e(title)	title in estimation output
e(tvar)	variable denoting time within groups
e(tmins)	formatted minimum time
e(tmaxs)	formatted maximum time
e(tsfmt)	format for the current time variable
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(method)	likelihood method
e(idencheck)	passed, failed, or skipped
e(technique)	maximization technique
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins

Matrices

e(b)	parameter vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(eigenvalues)	generalized eigenvalues
e(gradient)	gradient vector
e(shock_coeff)	estimated shock coefficient matrix
e(transition)	estimated state transition matrix
e(policy)	estimated policy matrix

`e(steady)`	estimated steady-state vector
`e(V)`	variance–covariance matrix of the estimators
`e(V_modelbased)`	model-based variance
Functions	
`e(sample)`	marks estimation sample

In addition to the above, the following is stored in `r()`:

Matrices	
`r(table)`	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in `r()` are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

A DSGE model is a system of equations in which the values of the control variables $\mathbf{y}_t$ and of the state variables $\mathbf{x}_t$ depend on their values in previous periods and their one-period-ahead rational expectations; see [DSGE] **Intro 1** for an introduction to control variables and state variables. The structural form of a nonlinear DSGE model is

$$E_t\left\{\mathbf{f}(\mathbf{x}_{t+1}, \mathbf{y}_{t+1}, \mathbf{x}_t, \mathbf{y}_t; \boldsymbol{\theta})\right\} = \mathbf{0} \tag{1}$$

where $\mathbf{f}$ is an $n \times 1$ vector of functions, the n_y variables $\mathbf{y}_t$ are control variables, and the $n_x \times 1$ variables $\mathbf{x}_t$ are state variables. The parameter $\boldsymbol{\theta}$ contains the model's structural parameters.

Fernández-Villaverde, Rubio-Ramírez, and Schorfheide (2016) contain an introduction to the solution and estimation of such models. `dsgenl` proceeds by linearizing (1) around the steady state and then solving the resulting linear system.

The steady state is found by solving the deterministic system of equations,

$$\mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{x}, \mathbf{y}; \boldsymbol{\theta}) = \mathbf{0}$$

which is n equations in n unknowns. The location of the steady state is a vector $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$, each element of which may depend on the structural parameters.

Once the steady state has been found, the vector of functions $\mathbf{f}$ is linearized around that point, yielding

$$E_t\left\{\mathbf{F}_1\mathbf{x}_{t+1} + \mathbf{F}_2\mathbf{y}_{t+1} + \mathbf{F}_3\mathbf{x}_t + \mathbf{F}_4\mathbf{y}_t\right\} = \mathbf{0} \tag{2}$$

Equation (2) is in the form of a linearized DSGE model, and the remaining solution techniques are the same as for `dsge`; see *Methods and formulas* in [DSGE] **dsge**. Parameter estimation proceeds on the basis of the linearized equation shown as (2).

Reference

Fernández-Villaverde, J., J. F. Rubio-Ramírez, and F. Schorfheide. 2016. Solution and estimation methods for DSGE models. In Vol. 2A of *Handbook of Macroeconomics*, ed. J. B. Taylor and H. Uhlig, chap. 9, 527–724. Amsterdam: North-Holland. https://doi.org/10.1016/bs.hesmac.2016.03.006.

Also see

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

[DSGE] **Intro 2** — Learning the syntax

[BAYES] **bayes: dsgenl** — Bayesian nonlinear dynamic stochastic general equilibrium models

[TS] **sspace** — State-space models

[TS] **tsset** — Declare data to be time-series data

[TS] **var** — Vector autoregressive models

[U] **20 Estimation and postestimation commands**

Title

dsgenl postestimation — Postestimation tools for dsgenl

Postestimation commands

The following postestimation commands are of special interest after `dsgenl`:

Command	Description
`estat covariance`	display model-implied covariance matrix of model variables
`estat policy`	display policy matrix of estimated model
`estat stable`	assess stability of the system
`estat steady`	display steady state of the system
`estat transition`	display transition matrix of estimated model
`irf`	create and analyze IRFs and FEVDs

The following standard postestimation commands are also available:

Command	Description
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estimates`	cataloging estimation results
`etable`	table of estimation results
`forecast`	dynamic forecasts and simulations
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of coefficients
`lrtest`	likelihood-ratio test
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients
`predict`	one-step-ahead predictions, prediction errors, and other diagnostic measures
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

predict

Description for predict

`predict` creates new variables containing predictions such as expected values. Predictions are available as static one-step-ahead predictions or as dynamic multistep predictions, and you can control when dynamic predictions begin.

Menu for predict

Statistics > Postestimation

Syntax for predict

`predict` [*type*] {*stub*`*`|*newvarlist*} [*if*] [*in*] [`,` *statistic options*]

statistic	Description
Main	
`xb`	linear prediction for observed variables
`states`	linear prediction for latent state variables

options	Description
Options	
`rmse(`*stub*`*`|*newvarlist*`)`	put estimated root mean squared errors of predicted statistics in new variables
`dynamic(`*time_constant*`)`	begin dynamic forecast at specified time
Advanced	
`smethod(`*method*`)`	method for predicting unobserved states

method	Description
`onestep`	predict using past information
`filter`	predict using past and contemporaneous information

Options for predict

Main

`xb`, the default, calculates the linear predictions of the observed variables.

`states` calculates the linear predictions of the latent state variables.

Options

`rmse(`*stub*`*` | *newvarlist*`)` puts the root mean squared errors of the predicted statistics into the specified new variables. The root mean squared errors measure the variances due to the disturbances but do not account for estimation error.

`dynamic(`*time_constant*`)` specifies when `predict` starts producing dynamic forecasts. The specified *time_constant* must be in the scale of the time variable specified in `tsset`, and the *time_constant* must be inside a sample for which observations on the dependent variables are available. For example, `dynamic(tq(2008q4))` causes dynamic predictions to begin in the fourth quarter of 2008, assuming that your time variable is quarterly; see [D] **Datetime**. If the model contains exogenous variables, they must be present for the whole predicted sample.

Advanced

`smethod(`*method*`)` specifies the method for predicting the unobserved states, `smethod(onestep)` and `smethod(filter)`, and causes different amounts of information on the dependent variables to be used in predicting the states at each time period.

`smethod(onestep)`, the default, causes `predict` to estimate the states at each time period using previous information on the dependent variables. The Kalman filter is performed on previous periods, but only the one-step predictions are made for the current period.

`smethod(filter)` causes `predict` to estimate the states at each time period using previous and contemporaneous data by the Kalman filter. The Kalman filter is performed on previous periods and the current period. `smethod(filter)` may be specified only with `states`.

Remarks and examples

For examples of `estat covariance`, see [DSGE] **Intro 3e** and [DSGE] **estat covariance**.

For examples of `estat policy`, see [DSGE] **Intro 1**, [DSGE] **Intro 3d**, and [DSGE] **Intro 3e**.

For examples of `estat transition`, see [DSGE] **Intro 1**, [DSGE] **Intro 3d**, and [DSGE] **Intro 3e**.

For examples of `irf` after `dsgenl`, see [DSGE] **Intro 3d** and [DSGE] **Intro 3e**.

For examples of `predict` and `forecast` after fitting a DSGE model, see [DSGE] **Intro 1**. Examples are shown with estimation results from `dsge` but apply to results from `dsgenl` as well.

Methods and formulas

The most important thing to keep in mind is that postestimation predictions after `dsgenl` are made on the basis of the linearized form of the model given to `dsgenl`.

Estimating the unobserved states is the key to predicting the observed variables.

By default and with the `smethod(onestep)` option, `predict` estimates the state in each period by applying the Kalman filter to all previous periods and only making the one-step prediction to the current period.

With the `smethod(filter)` option, `predict` estimates the states in each period by applying the Kalman filter on all previous periods and the current period. The computational difference between `smethod(onestep)` and `smethod(filter)` is that `smethod(filter)` performs the update step on the current period while `smethod(onestep)` does not. The statistical difference between `smethod(onestep)` and `smethod(filter)` is that `smethod(filter)` uses contemporaneous information on the observed variables while `smethod(onestep)` does not.

The observed control variables are predicted by plugging in the estimated states.

Also see

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

[DSGE] **estat covariance** — Display estimated covariances of model variables

[DSGE] **estat policy** — Display policy matrix

[DSGE] **estat stable** — Check stability of system

[DSGE] **estat steady** — Display steady state of nonlinear DSGE model

[DSGE] **estat transition** — Display state transition matrix

[TS] **forecast** — Econometric model forecasting

[TS] **irf** — Create and analyze IRFs, dynamic-multiplier functions, and FEVDs

[U] 20 Estimation and postestimation commands

Title

estat covariance — Display estimated covariances of model variables

Description

`estat covariance` displays model-implied covariances between control variables.

Quick start

Display variances and covariances of all control variables in the model

```
estat covariance
```

Include only variances and covariances of `x` and its lag

```
estat covariance x L.x
```

Display the variance of `x1` and the covariance of `x1` with the lag of `x1` and with `x2`

```
estat covariance x1, addcovariance(L.x1 x2)
```

Menu for estat

Statistics > Postestimation

Syntax

`estat covariance` [*varlist*] [, *options*]

varlist may include control variables and their lags. If *varlist* is not specified, variances and covariances are reported for all control variables in the model.

options	Description
`addcovariance(`*clistc*`)`	report additional covariances
`nocovariance`	do not report covariances
`post`	post variances and covariances and their VCE as estimation results
`level(#)`	set confidence level; default is `level(95)`
display_options	control columns and column formats and line width

`collect` is allowed; see [U] **11.1.10 Prefix commands**.

Options

`addcovariance(`*clistc*`)` specifies that the covariances between the control variables specified in *clistc* and those specified in *varlist* be displayed. The variances of variables in *clistc* are not reported. *clistc* can contain lags of the control variables in the model.

`nocovariance` specifies that no covariance be displayed. `nocovariance` may not be specified with `addcovariance()`.

`post` causes `estat covariance` to behave like a Stata estimation (e-class) command. `estat covariance` posts the estimated variance–covariance matrix to `e()`, so you can treat it just as you would results from any other estimation command.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see **[U] 20.8 Specifying the width of confidence intervals**.

display_options: `noci`, `nopvalues`, `cformat(%fmt)`, `pformat(%fmt)`, `sformat(%fmt)`, and `nolstretch`; see [R] **Estimation options**.

Remarks and examples

`estat covariance` displays covariances between control variables implied by a DSGE model.

We provide examples of the use of `estat covariance` using the model from [DSGE] **Intro 3e**. In that introduction, we estimated some of the parameters of a Real Business Cycle model. Model setup and estimation were given by

```
. use https://www.stata-press.com/data/r18/usmacro2
(Federal Reserve Economic Data - St. Louis Fed, 2017-01-15)
. constraint 1 _b[alpha] = 0.33
. constraint 2 _b[beta]  = 0.99
. constraint 3 _b[delta] = 0.025
. constraint 4 _b[chi]   = 2

. dsgenl (1/c = {beta}*(1/F.c)*(1+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) nolog
Solving at initial parameter vector ...
Checking identification ...
First-order DSGE model
Sample: 1955q1 thru 2015q4                              Number of obs = 244
Log likelihood = -639.38787
 ( 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	.3132838	.0614709	5.10	0.000	.192803	.4337646
sd(e.z)	2.249022	.101853			2.049394	2.44865

The commands just listed imported data, set up constraints, and estimated the remaining model parameters. For details of the model, see [DSGE] **Intro 3e**. The control variables in this model are `c`, `i`, `r`, `w`, `h`, and `y`.

By default, `estat covariance` displays the contemporaneous variances and covariances of all the control variables in the model.

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

	Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
c						
var(c)	.8569582	.1674995	5.12	0.000	.5286652	1.185251
cov(c,i)	1.700528	.4329933	3.93	0.000	.8518764	2.549179
cov(c,r)	-.4155731	.0553716	-7.51	0.000	-.5240994	-.3070467
cov(c,w)	.9560928	.198436	4.82	0.000	.5671653	1.34502
cov(c,h)	.0991346	.031423	3.15	0.002	.0375466	.1607226
cov(c,y)	1.055227	.2295048	4.60	0.000	.6054062	1.505049
i						
var(i)	208.0512	20.07879	10.36	0.000	168.6975	247.4049
cov(i,r)	50.26689	4.824216	10.42	0.000	40.8116	59.72218
cov(i,w)	25.95045	2.626602	9.88	0.000	20.8024	31.09849
cov(i,h)	24.24992	2.32649	10.42	0.000	19.69008	28.80975
cov(i,y)	50.20036	4.943248	10.16	0.000	40.51178	59.88895
r						
var(r)	12.95503	1.275345	10.16	0.000	10.4554	15.45466
cov(r,w)	5.540528	.5242116	10.57	0.000	4.513092	6.567963
cov(r,h)	5.956101	.5726953	10.40	0.000	4.833639	7.078563
cov(r,y)	11.49663	1.096581	10.48	0.000	9.34737	13.64589
w						
var(w)	3.893379	.4600228	8.46	0.000	2.991751	4.795007
cov(w,h)	2.937286	.29054	10.11	0.000	2.367838	3.506734
cov(w,y)	6.830665	.7434327	9.19	0.000	5.373564	8.287766
h						
var(h)	2.838151	.2712547	10.46	0.000	2.306502	3.369801
cov(h,y)	5.775437	.5612467	10.29	0.000	4.675414	6.875461
y						
var(y)	12.6061	1.297192	9.72	0.000	10.06365	15.14855

Applied studies frequently look at the variances and covariances of a subset of the control variables in the model. Studies also frequently look at the covariances between a subset of the control variables in the model and the lags of one or two control variables. We illustrate how to obtain these results.

To view only the variance of the consumption c and its covariance with the control variables in the model, type

```
. estat covariance c, addcovariance(i r w h y)
Estimated covariances of model variables
```

	Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
c						
var(c)	.8569582	.1674995	5.12	0.000	.5286652	1.185251
cov(c,i)	1.700528	.4329933	3.93	0.000	.8518764	2.549179
cov(c,r)	-.4155731	.0553716	-7.51	0.000	-.5240994	-.3070467
cov(c,w)	.9560928	.198436	4.82	0.000	.5671653	1.34502
cov(c,h)	.0991346	.031423	3.15	0.002	.0375466	.1607226
cov(c,y)	1.055227	.2295048	4.60	0.000	.6054062	1.505049

Autocovariances are available as well. To view the variance of output and the first-order autocovariance of output, type

```
. estat covariance y, addcovariance(L.y)
Estimated covariances of model variables
```

	Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
y						
var(y)	12.6061	1.297192	9.72	0.000	10.06365	15.14855
cov(y,L.y)	4.370091	1.077205	4.06	0.000	2.258808	6.481374

To view the variances and covariance of output and consumption as well as their covariances with lagged output and lagged consumption, type

```
. estat covariance y c, addcovariance(L.y L.c)
Estimated covariances of model variables
```

	Coefficient	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
y						
var(y)	12.6061	1.297192	9.72	0.000	10.06365	15.14855
cov(y,c)	1.055227	.2295048	4.60	0.000	.6054062	1.505049
cov(y,L.y)	4.370091	1.077205	4.06	0.000	2.258808	6.481374
cov(y,L.c)	1.000102	.2569841	3.89	0.000	.4964225	1.503782
c						
var(c)	.8569582	.1674995	5.12	0.000	.5286652	1.185251
cov(y,c)	1.055227	.2295048	4.60	0.000	.6054062	1.505049
cov(c,L.y)	1.000102	.1561667	6.40	0.000	.694021	1.306183
cov(c,L.c)	.842517	.1656435	5.09	0.000	.5178617	1.167172

What we typed requested the covariance between y and the lag of c (cov(y,L.c)) and between c and the lag of y (cov(c,L.y)). In general, these covariances can be different. The structure in this model causes their estimates to be the same, but their estimated standard errors are different. The estimated standard errors differ because the derivatives that enter the delta method are different.

Stored results

`estat covariance` stores the following in `r()`:

Matrices

`r(b)`	estimates
`r(V)`	variance–covariance matrix of the estimates

If `post` is specified, `estat covariance` also stores the following in `e()`:

Macros

`e(properties)`	`b V`

Matrices

`e(b)`	estimates
`e(V)`	variance–covariance matrix of the estimates

Methods and formulas

Entries in the covariance matrix are functions of the structural parameter vector θ. Standard errors are calculated using the delta method.

Also see

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

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

[DSGE] **Intro 3e** — Nonlinear New Classical model

Title

estat policy — Display policy matrix

Description

`estat policy` displays the estimated policy matrix of the state-space form of a DSGE model.

Quick start

Display the estimated policy matrix

```
estat policy
```

Same as above, but with 90% confidence intervals

```
estat policy, level(90)
```

Menu for estat

Statistics > Postestimation

Syntax

```
estat policy [, compact post level(#) display_options]
```

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

Options

`compact` reports only the coefficient values of the estimated policy matrix and displays these coefficients in matrix form.

`post` causes `estat policy` to behave like a Stata estimation (e-class) command. `estat policy` posts the policy matrix parameters along with the estimated variance–covariance matrix to `e()`, so you can treat the estimated policy matrix as you would results from any other estimation command.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see **[U] 20.8 Specifying the width of confidence intervals**.

display_options: `noci`, `nopvalues`, `cformat(%fmt)`, `pformat(%fmt)`, `sformat(%fmt)`, and `nolstretch`; see [R] **Estimation options**.

Remarks and examples

The policy matrix is part of the state-space form of a DSGE model. It specifies the model's control variables as a function of the model's state variables.

For examples, see [DSGE] **Intro 1**, [DSGE] **Intro 3a**, and [DSGE] **Intro 3c**.

Stored results

`estat policy` stores the following in `r()`:

Matrices

`r(policy)`	estimated policy matrix
`r(b)`	estimates
`r(V)`	variance–covariance matrix of the estimates

If `post` is specified, `estat policy` also stores the following in `e()`:

Macros

`e(properties)`	`b V`

Matrices

`e(policy)`	estimated policy matrix
`e(b)`	estimates
`e(V)`	variance–covariance matrix of the estimates

Methods and formulas

Entries in the policy matrix $\mathbf{G}$ are functions of the structural parameter vector $\boldsymbol{\theta}$. Standard errors for $\widehat{\mathbf{G}}$ are calculated using the delta method.

Also see

[DSGE] **dsge** — Linear dynamic stochastic general equilibrium models

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

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

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

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

Title

estat stable — Check stability of system

Description

`estat stable` displays functions of the model parameters that indicate whether the model is saddle-path stable at specific parameter values. These results can help you find initial values for which the model is saddle-path stable. Saddle-path stability is required for solving and estimating the parameters of DSGE models.

Menu for estat

Statistics > Postestimation

Syntax

```
estat stable
```

`collect` is allowed; see [U] **11.1.10 Prefix commands**.

Remarks and examples

DSGE models are dynamic systems that are subject to random shocks. DSGE models that do not spiral out of control or converge to a single point when shocked are said to be "saddle-path stable". We can solve for the state-space form of DSGE models only if they are saddle-path stable, and we can estimate the parameters of models only if we can solve for the state-space form.

The structural parameter values determine whether a DSGE model is saddle-path stable. In the process of solving the structural form for the state-space form, the Klein (2000) solver computes the generalized eigenvalues of a matrix formed from structural parameter values. An eigenvalue is said to be stable when its absolute value is less than 1. The model is saddle-path stable when the number of stable eigenvalues equals the number of states in the model.

`estat stable` displays the generalized eigenvalues implied by the parameter values in `e(b)`, indicates which are stable and which are unstable, and displays a note indicating whether the model is saddle-path stable at these parameter values. `estat stable` can help you find initial values for the maximization routine when the default values imply a model that is not saddle-path stable; see [DSGE] **Intro 5** for details.

Stored results

`estat stable` stores the following in `r()`:

Scalars

`r(stable)`	`1` if stable, `0` otherwise

Matrices

`r(eigenvalues)`	eigenvalues

Methods and formulas

`estat stable` displays the generalized eigenvalues computed by the Klein (2000) solver. Values less than 1 are labeled as stable; values greater than or equal to 1 are labeled as unstable.

Reference

Klein, P. 2000. Using the generalized Schur form to solve a multivariate linear rational expectations model. *Journal of Economic Dynamics and Control* 24: 1405–1423. https://doi.org/10.1016/S0165-1889(99)00045-7.

Also see

[DSGE] **dsge** — Linear dynamic stochastic general equilibrium models

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

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

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

[DSGE] **Intro 5** — Stability conditions

Title

estat steady — Display steady state of nonlinear DSGE model

Description

`estat steady` displays the estimated steady-state values of variables in a nonlinear DSGE model.

Quick start

Display the estimated steady-state values

```
estat steady
```

Same as above, but with 90% confidence intervals

```
estat steady, level(90)
```

Menu for estat

Statistics > Postestimation

Syntax

```
estat steady [ , compact level(#) display_options ]
```

`collect` is allowed; see [U] **11.1.10 Prefix commands**.

Options

`compact` reports only the coefficient values of the estimated steady-state vector.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [U] **20.8 Specifying the width of confidence intervals**.

display_options: `noci`, `nopvalues`, `cformat(%fmt)`, `pformat(%fmt)`, `sformat(%fmt)`, and `nolstretch`; see [R] **Estimation options**.

Remarks and examples

A nonlinear DSGE model is a system of equations describing the relationships among model variables. These relationships are dynamic and subject to random shocks. When the random shocks are set to zero, the model variables converge to fixed values over time. This vector of fixed values is known as the steady-state vector. The steady-state vector solves the system of equations when past and future values of each variable are replaced with their present value and expectations are dropped.

`estat steady` displays the estimated steady-state value of each variable.

For an example of `estat steady`, see [DSGE] **Intro 3f**.

Stored results

`estat steady` stores the following in `r()`:

Matrices

`r(steady)`	estimated steady-state vector

Methods and formulas

A nonlinear DSGE model can be written in the structural form

$$E_t\left\{\mathbf{f}(\mathbf{x}_{t+1},\mathbf{y}_{t+1},\mathbf{x}_t,\mathbf{y}_t;\boldsymbol{\theta})\right\}=\mathbf{0}$$

The steady state is obtained by solving

$$\mathbf{f}(\mathbf{x},\mathbf{y},\mathbf{x},\mathbf{y};\boldsymbol{\theta})=\mathbf{0}$$

The resulting $(\mathbf{x},\mathbf{y})$ is a function of the parameter vector $\boldsymbol{\theta}$. Standard errors are calculated using the delta method.

Also see

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

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

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

Title

estat transition — Display state transition matrix

Description

`estat transition` displays the estimated state transition matrix of the state-space form of a DSGE model.

Quick start

Display the estimated transition matrix

```
estat transition
```

Same as above, but with 90% confidence intervals

```
estat transition, level(90)
```

Menu for estat

Statistics > Postestimation

Syntax

```
estat transition [, compact post level(#) display_options]
```

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

Options

`compact` reports only the coefficient values of the estimated policy matrix and displays these coefficients in matrix form.

`post` causes `estat transition` to behave like a Stata estimation (e-class) command. `estat transition` posts the state transition matrix to `e()`, so you can treat it as you would results from any other estimation command.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see **[U] 20.8 Specifying the width of confidence intervals**.

display_options: `noci`, `nopvalues`, `cformat(%fmt)`, `pformat(%fmt)`, `sformat(%fmt)`, and `nolstretch`; see [R] **Estimation options**.

Remarks and examples

The state transition matrix is part of the state-space form of a DSGE model. It specifies the transition matrix of the model's state variables.

For examples, see [DSGE] **Intro 1**, [DSGE] **Intro 3a**, and [DSGE] **Intro 3b**.

Stored results

`estat transition` stores the following in `r()`:

Matrices

`r(transition)`	estimated transition matrix
`r(b)`	estimates
`r(V)`	variance–covariance matrix of the estimates

If `post` is specified, `estat transition` also stores the following in `e()`:

Macros

`e(properties)`	`b V`

Matrices

`e(transition)`	estimated transition matrix
`e(b)`	estimates
`e(V)`	variance–covariance matrix of the estimates

Methods and formulas

Entries in the state transition matrix $\mathbf{H}$ are functions of the structural parameter vector $\boldsymbol{\theta}$. Standard errors for entries in $\widehat{\mathbf{H}}$ are calculated using the delta method.

Also see

[DSGE] **dsge** — Linear dynamic stochastic general equilibrium models

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

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

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

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

Glossary

autoregressive process. An autoregressive process is a time series in which the current value of a variable is a linear function of its own past values and a white-noise error term. A first-order autoregressive process, denoted as an AR(1) process, is $y_t = \rho y_{t-1} + \epsilon_t$. An AR($p$) model contains p lagged values of the dependent variable.

conditional mean. A conditional mean expresses the mean of one variable as a function of some other variables. A regression function is a conditional mean.

control variable. A control variable is an endogenous variable. Control variables can be observed or unobserved.

In a structural DSGE model, the current value of a control variable depends on the current value of other control variables, on the expected future values of any control variable, and on the current values of state variables. The current value of a control variable is found by solving the model for the state-space form.

covariance stationary. A covariance stationary process is a weakly stationary process.

dynamic forecast. A dynamic forecast uses forecasted values wherever lagged values of the endogenous variables appear in the model, allowing one to forecast multiple periods into the future.

dynamic stochastic general equilibrium model. A dynamic stochastic general equilibrium model is a multivariate time-series model that specifies the structural relationship between state variables and control variables and is typically derived from economic theory.

endogenous variable. An endogenous variable is a variable whose values are determined by the equilibrium of a structural model. The values of an endogenous variable are determined inside the system.

equilibrium. The equilibrium values of variables in a model are the values that satisfy all the model's equations simultaneously.

exogenous variable. An exogenous variable is one whose values change independently of the other variables in a structural model. The values of an exogenous variable are determined outside the system. In a time-series model, an exogenous variable is also a predetermined variable.

expected future value. An expected future value is a forecast of the value of a variable in the future based on current information. In DSGE models, expected future values are computed under rational expectations.

Under rational expectations, $E_t(y_{t+1})$ is the condition mean of y_{t+1} conditional on the complete history of all variables in the model and the structure of the model itself.

forward operator. The forward operator F denotes the value of a variable at time $t+1$. Formally, $Fy_t = y_{t+1}$, and $F^2 y_t = Fy_{t+1} = y_{t+2}$. A forward operator is also called a lead operator.

identified. Identified is a condition required to estimate the parameters of a model. In other words, only identified parameters can be estimated.

In DSGE models, the parameters are identified when there is a unique parameter vector that maximizes the likelihood function. For a discussion of identification, see [DSGE] **Intro 6**.

impulse–response function. An impulse–response function (IRF) measures the effect of a shock to an endogenous variable on itself or another endogenous variable. The kth impulse–response function of variable i on variable j measures the effect on variable j in period $t+k$ in response to a one-unit shock to variable i in period t, holding everything else constant.

independent and identically distributed. A series of observations is independent and identically distributed (i.i.d.) if each observation is an independent realization from the same underlying distribution. In some contexts, the definition is relaxed to mean only that the observations are independent and have identical means and variances; see Davidson and MacKinnon (1993, 42).

initial values. Initial values specify the starting place for the iterative maximization algorithm used by DSGE.

Kalman filter. The Kalman filter is a recursive procedure for predicting the state vector in a state-space model.

lag operator. The lag operator L denotes the value of a variable at time $t-1$. Formally, $Ly_t = y_{t-1}$, and $L^2 y_t = Ly_{t-1} = y_{t-2}$.

lead operator. See *forward operator*.

likelihood-ratio (LR) test. The LR test is a classical testing procedure used to compare the fit of two models, one of which, the constrained model, is nested within the full (unconstrained) model. Under the null hypothesis, the constrained model fits the data as well as the full model. The LR test requires one to determine the maximal value of the log-likelihood function for both the constrained and the full models.

linearized model. A linearized model is an approximation to a model that is nonlinear in the variables and nonlinear in the parameters. The approximation is linear in variables but potentially nonlinear in the parameters. In a linearized model, variables are interpreted as unit deviations from steady state.

log-linear model. A log-linear model is an approximation to a model that is nonlinear in the variables and nonlinear in the parameters. In a log-linear model, variables are interpreted as percentage deviations from steady state.

model solution. A model solution is a function for the endogenous variables in terms of the exogenous variables. A model solution is also known as the reduced form of a model.

In DSGE terminology, a model solution expresses the control variables as a function of the state variables alone and expresses the state variables as a function of their values in the previous period and shocks. The reduced form of a DSGE model is also known as the state-space form of the DSGE model.

model-consistent expectation. A model-consistent expectation is the conditional mean of a variable implied by the model under consideration.

For example, under rational expectations the model-consistent expectation of $E_t(y_{t+1})$ is the mean of y_{t+1} implied by the model, conditional on the realizations of variables dated time t or previously.

nonpredetermined variable. A nonpredetermined variable is a variable whose value at time t is determined by the system of equations in the model. Contrast with predetermined variable.

null hypothesis. In hypothesis testing, the null hypothesis typically represents the conjecture that one is attempting to disprove. Often the null hypothesis is that a parameter is zero or that a statistic is equal across populations.

one-step-ahead forecast. See *static forecast*.

policy matrix. The policy matrix in the reduced form of a DSGE model is the matrix that expresses control variables as a function of state variables.

predetermined variable. A predetermined variable is a variable whose value is fixed at time t, given everything that has occurred previously. More technically, the value of a predetermined variable is fixed, given the realizations of all observed and unobserved variables at times $t-1$, $t-2$, ...

rational expectations. A rational expectation of a variable does not deviate from the mean of that variable in a predictable way. More technically, a rational expectation of a variable is the conditional mean of the variable implied by the model.

realization. The realization of a random variable is the value it takes on when drawn.

reduced form. The reduced form of a model expresses the endogenous variables as functions of the exogenous variables.

The reduced form of a DSGE model expresses the control variables as a function of the state variables alone and expresses the state variables as a function of their values in the previous period and shocks. The reduced form of a DSGE model is a state-space model.

saddle-path stable. A saddle-path stable model is a structural model that can be solved for its state-space form. The existence of a saddle-path stable solution depends on the parameter values of the model. For a discussion of saddle-path stability, see [DSGE] **Intro 5**.

shock variable. A shock variable is a random variable whose value is specified as an independently and identically distributed (i.i.d.) random variable. The maximum likelihood estimator is derived under normally distributed shocks but remains consistent under i.i.d. shocks. Robust standard errors must be specified when the errors are i.i.d. but not normally distributed.

state transition matrix. The state transition matrix in the reduced form of a DSGE model is the matrix that expresses how the future values of state variables depend on their current values.

state variable. A state variable is an unobserved exogenous variable.

In DSGE models, a state variable is an unobserved exogenous variable that may depend on its own previous value, the previous values of other state variables, and shocks.

state-space model. A state-space model describes the relationship between an observed time series and an unobservable state vector that represents the "state" of the world. The measurement equation expresses the observed series as a function of the state vector, and the transition equation describes how the unobserved state vector evolves over time. By defining the parameters of the measurement and transition equations appropriately, one can write a wide variety of time-series models in the state-space form.

For DSGE models, the state-space form is the reduced form of the structural model.

The DSGE framework changes the jargon and the structure of state-space models. The measurement equation is the vector of equations for the control variables, and the transition equation is the vector of equations for the state variables. In contrast to the standard state-space model, DSGE models allow a control variable to be unobserved.

static forecast. A static forecast uses actual values wherever lagged values of the endogenous variables appear in the model. As a result, static forecasts perform at least as well as dynamic forecasts, but static forecasts cannot produce forecasts into the future when lags of the endogenous variables appear in the model.

Because actual values will be missing beyond the last historical time period in the dataset, static forecasts can forecast only one period into the future (assuming only first lags appear in the model); thus they are often called one-step-ahead forecasts.

steady-state equilibrium. A steady-state equilibrium is a time-invariant rest point of a dynamic system.

More technically, a steady-state equilibrium is a set of values for the endogenous variables to which the dynamic system will return after an exogenous variable is changed or a random shock occurs. This set of values is time invariant in that it does not depend on the time period in which the change or shock occurs. Multistep dynamic forecasts converge to these values. A steady-state

equilibrium is also known as a long-run equilibrium because it specifies time-invariant values for the endogenous variables to which the dynamic system will return, if left unshocked.

stochastic equation. A stochastic equation, in contrast to an identity, is an equation in a forecast model that includes a random component, most often in the form of an additive error term. Stochastic equations include parameters that must be estimated from historical data.

stochastic trend. A stochastic trend is a nonstationary random process. Unit-root process and random coefficients on time are two common stochastic trends. See [TS] **ucm** for examples and discussions of more commonly applied stochastic trends.

strict stationarity. A process is strictly stationary if the joint distribution of $y_1, \ldots, y_k$ is the same as the joint distribution of $y_{1+\tau}, \ldots, y_{k+\tau}$ for all k and τ. Intuitively, shifting the origin of the series by τ units has no effect on the joint distributions.

structural model. A structural model specifies the theoretical relationship among a set of variables. Structural models contain both endogenous variables and exogenous variables. Parameter estimation and interpretation require that structural models be solved for a reduced form.

trend. The trend specifies the long-run behavior in a time series. The trend can be deterministic or stochastic. Many economic, biological, health, and social time series have long-run tendencies to increase or decrease. Before the 1980s, most time-series analysis specified the long-run tendencies as deterministic functions of time. Since the 1980s, the stochastic trends implied by unit-root processes have become a standard part of the toolkit.

Wald test. A Wald test is a classical testing procedure used to compare the fit of two models, one of which, the constrained model, is nested within the full (unconstrained) model. Under the null hypothesis, the constrained model fits the data as well as the full model. The Wald test requires one to fit the full model but does not require one to fit the constrained model.

weakly stationary. A process is weakly stationary if the mean of the process is finite and independent of t, the unconditional variance of the process is finite and independent of t, and the covariance between periods t and $t - s$ is finite and depends on $t - s$ but not on t or s themselves. Weakly-stationary processes are also known as covariance stationary processes.

white noise. A variable u_t represents a white-noise process if the mean of u_t is zero, the variance of u_t is σ^2, and the covariance between u_t and u_s is zero for all $s \neq t$.

Reference

Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.

Subject and author index

See the combined subject index and the combined author index in the *Stata Index*.