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
Syntax for linear DSGE models
  Preview of dsge syntax
  Specifying the system of linear equations
  Control variables
  State variables and shocks
  Expectations of future values of control variables
  Specifying parameters using dsge's substitutable expressions
Syntax for nonlinear DSGE models
  Preview of dsgenl syntax
  Specifying the system of nonlinear equations
  State and control variables
  Expectations in nonlinear models

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 = exp*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 = exp*F.y + exp*z + exp*x)          ///
  (F.x = exp*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 = exp*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