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/r19/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	.2666191	2.53	0.012	.1509833	1.196111
rho_z2	.2710012	.2564752	1.06	0.291	-.231681	.7736834
sd(e.u)	2.315264	.29922			1.728803	2.901724
sd(e.z)	.7720676	.171628			.4356829	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