Create and Parameterize Model Object

read_model.m

20 October 2016

Introduction

Read in the model file, `endo_cred_v3.model`, and create two model objects: one with a policy reaction function, the other with optimal commitment policy derived from a loss function. Parameterize the model objects, find their steady states, and first-order solution matrices. Save the model objects for further use in other m-files.

Contents

1 Clear Workspace

Clear workspace (system memory) of all variables (1), close all existing figure windows (2), and clear command window (3).

```
clear; (1)
close all; (2)
clc; (3)
warning('off','IRIS:userdataobj:export');
%#ok<*NOPTS>
```

2 Create Model with Reaction Function

Use the function `model(...)` (4) to load the model file, `endo_cred_v3.model`, and create a model object named `mr`. When creating the model object, set the following control parameters used in the model file: `'SW_ENDO_CRED=' false` (5) and `'SW_OPTIMAL=' false` (6). This means that the model objects will be based on equations with endogenous credibility and a simple policy reaction function (see the model file `endocredv2.model`).

```
mr = model( 'endo_cred_v3.model', ... (4)
            'SW_NONLIN_PC',true, ...
            'SW_ENDO_CRED=',true, ... (5)
            'SW_OPTIMAL=',false) (6)
```

3 Parameterize Model Object with Reaction Function

When a new model object is created, its parameters are set to `NaN` by default (7). To assign the parameters, use the dot-name syntax (8). Ignore the values after the comment sign, % (9); these were just part of some earlier parameterizations.

```
get(mr,'parameters') (7)
```

Output gap equation parameters.

```
mr.alp1 = 0.60; (8)
mr.alp2 = 0.07;
% mr.alp3 = 0.05; (9)
mr.alp3 = 0.08;
mr.alp4 = 0.05;
```

Core inflation equation (Phillips curve) parameters.

```
56 mr.bet1 = 0.666667;
57 mr.bet2 = 0.06;
58 mr.bet3 = 0.005;
59 mr.the1 = 0.6;
```

Phillips curve nonlinearity parameters.

```
64 mr.iot1 = 0.4;
```

Policy reaction function parameters.

```
69 mr.gam1 = 0.75;
70 mr.gam2 = 3;
```

Foreign exchange equation parameters.

```
75 mr.del1 = 0.5;
76 mr.del2 = 0.6;
77 mr.del11 = 0.3;
```

Bias parameters.

```
82 mr.bp = 0.25;
83 mr.bs = 0;
```

Credibility parameters.

```
88 mr.eta1 = 0.80;
89 mr.eta2 = 1;
90 mr.hi = 8;
```

Steady state parameters.

```
95 mr.pi = 4;
96 mr.C = 1;
97 mr.rho = 2;
```

Verify that all parameters are assigned using the function `get(...)`.

```
103 get(mr,'parameters');
```

4 Find Steady State and Compute First-Order Matrices

After parameterizing the model object with a reaction function, `mr`, find its steady state using the function `sstate(...)` with two options. The option `'growth=' true` means that some variables may be growing at a constant rate in steady state (along a balanced growth path), as opposed to a default stationary steady state. The option `'blocks=' true` means that the model equations will be analyzed and split up into recursive blocks to make the convergence faster and easier. The function `sstate(...)` relies on Optimization Toolbox routines. Verify that the steady state is really a valid one (a), and display the steady state levels (b) and growth rates (c) (i.e. period-by-period differences).

With a valid steady state assigned, run the function `solve(...)` (d) to compute the first-order solution matrices. These matrices will be made use of even in nonlinear simulations of the model. When displaying the model object on the screen (no semicolon at the end of the last command in this section), the message now includes a reference to existing solutions.

```
126 mr = sstate(mr, ...
127     'growth=',true, ...
128     'blocks=',true);
129
130 chksstate(mr); (a)
131 get(mr,'sstateLevel') (b)
132 get(mr,'sstateGrowth') (c)
133
134 mr = solve(mr) (d)
```

5 Create Model Object with Optimal Commitment and Nonlin PC

Load the same model file again, now with the control parameter `'SW_OPTIMAL=' true` (e); this means that the resulting model object, `mo`, will be based on equations with a loss function, see again the model file `endo_cred_v3.model`. Use the option `'optimal='` (f) to specify what kind of optimal policy is to be calculated (here: optimal commitment). In model objects with optimal policy, new variables (for the lagrange multipliers) and equations derived from the loss function are automatically created and added to the model; display all names (variables, parameters, shocks) (g) and all equations using the function `get(...)` (h).

```
149 mon = model('endo_cred_v3.model', ...
150             'SW_NONLIN_PC=',true, ...
151             'SW_ENDO_CRED=',true, ...
152             'SW_OPTIMAL=',true, ... (e)
153             'optimal=',{'type=','commitment'} ) (f)
154
155 get(mon,'names')' (g)
156 get(mon,'equations')' (h)
```

6 Parameterize and Solve Model with Optimal Commitment

Assign the same parameters as in the model with a reaction function, `mr`; to do so in a single command line, combine the functions `assign(...)` and `get(...)` (i). Some parameters in the new model object, `mon`, are still unassigned: those describing the loss function (j). Complete the calibration by assigning them using the dot-name syntax again (k).

Find the steady state of the new model with optimal policy (l) setting the option `'zeroMultipliers='true` (m) to indicate that all multipliers associated with the loss function can be set to zero and excluded from numerical iterations. The steady state of the model object with optimal policy is identical to the steady state of the model object with a reaction function (n).

Finally, compute the first-order solution matrices for `mon`.

```
mon = assign( mon, get(mr,'sstate') ); (i)

get(mon,'parameters') (j)
```

Assign loss function parameters.

```
mon.psi = 0.98; (k)
mon.lmb1 = 1;
mon.lmb2 = 1;
mon.lmb3 = 0.5;

mon = sstate(mon, ... (l)
    'growth=',true, ...
    'blocks=',true, ...
    'zeroMultipliers=',true ); (m)

chksstate(mon);
get(mr,'sstateLevel') & get(mon,'sstateLevel') (n)

mon = solve(mon)
```

7 Create Model Object with Optimal Commitment and Linearized PC

```
mol = model('endo_cred_v3.model', ...
            'SW_NONLIN_PC=',false, ...
            'SW_ENDO_CRED=',true, ...
            'SW_OPTIMAL=',true, ... (e)
            'optimal=',{'type=','commitment'} ) (f)

mol = assign(mol,mon);
chksstate(mol);
```

```
209 mol = solve(mol);
```

8 Save Model Objects to MAT File for Further Use

Save the two parameterized and solved model objects, `mr` (reaction function) and `mo` (optimal commitment) to a MAT file (Matlab binary file) named `read_model.mat` for further access and use in other m-files.

```
217 save read_model.mat mon mol;
```

9 Help on IRIS Functions Used in This File

Use either `help` to display help in the command window, or `idoc` to display help in an HTML browser window.

```
idoc model/Contents
idoc model/model
idoc model/get
idoc model/subsasgn
idoc model/sstate
idoc model/chksstate
idoc model/solve
idoc model/assign
```