Matlab: R2015a IRIS: 20150527

Assign and Change Parameters and Steady States

change_parameters_and_sstates.m

by Jaromir Benes

May 27, 2015

Summary

Assign or change the values of parameters and/or steady states of variables in a model object using a number of different ways. Under different circumstances, different methods of assigning parameters may be more convenient (but they, of course, all do the same job).

Contents

1 Clear Workspace

Clear workspace, close all graphics figures, clear command window, and check the IRIS version.

```
clear;
close all;
clc;
irisrequired 20140315;
%#ok<*NOPTS>
%#ok<*NASGU>
```

2 Read Model File and Assign Parameters to Model Object

The easiest way to assign or change parameters is simply by using the dot-reference, i.e. the name of the model object dot the name of the parameter [1].

```
m = model('simple_SPBC.model');

m.alpha = 1.03^(1/4); [1]
m.beta = 0.985^(1/4);
m.gamma = 0.60;
m.delta = 0.03;
m.pi = 1.025^(1/4);
m.eta = 6;
m.k = 10;
m.psi = 0.5;

m.chi = 0.80;
m.xiw = 60;
m.xip = 80;
m.rhoa = 0.90;

m.rhor = 0.8;
m.kappap = 2.5;
m.kappan = 0.1;

m.Short_ = 0;
m.Wage_ = 0;

m.std_Mp = 0;
m.std_Mw = 0;
m.std_Ea = 0.1/100;
```

3 Assign Parameter Database When Reading Model File

Create first a database with the desired parameter values [2] (or use an existing one, for example), and assign the database when reading the model file, i.e. when calling the function `model` [3], by using the option `assign=`.

```
P = struct();

P.alpha = 1.03^(1/4); [2]
P.beta = 0.985^(1/4);
P.gamma = 0.60;
P.delta = 0.03;
P.pi = 1.025^(1/4);
P.eta = 6;
P.k = 10;
P.psi = 0.5;

P.chi = 0.80;
P.xiw = 60;
P.xip = 80;
P.rhoa = 0.90;

P.rhor = 0.8;
P.kappap = 2.5;
P.kappan = 0.1;

P.Short_ = 0;
P.Wage_ = 0;

P.std_Mp = 0;
P.std_Mw = 0;
P.std_Ea = 0.1/100;

m = model(’simple_SPBC.model’, ...
    ’assign=’,P); [3]
```

4 Åssign Parameter Database After Reading Model File

Here, use again a parameter database, but assign the database after reading the model file, in a separate call to the function `assign` [4].

```
P = struct();

```

```
P.alpha = 1.03^(1/4);
P.beta = 0.985^(1/4);
P.gamma = 0.60;
P.delta = 0.03;
P.pi = 1.025^(1/4);
P.eta = 6;
P.k = 10;
P.psi = 0.5;

P.chi = 0.80;
P.xiw = 60;
P.xip = 80;
P.rhoa = 0.90;

P.rhor = 0.8;
P.kappap = 2.5;
P.kappan = 0.1;

P.Short_ = 0;
P.Wage_ = 0;

P.std_Mp = 0;
P.std_Mw = 0;
P.std_Ea = 0.1/100;

m = model('simple_SPBC.model');

m = assign(m,P); [4]>
```

5 Change Parameters in Model Object

There are several ways how to change some of the parameters. All the following three blocks of code do exactly the same.

Refer directly to the model object using a model-dot-name notation.

```
m.chi = 0.9;
m.xip = 100;
```

Use the function assign and specify name-value pairs; you can optionally use the equal signs [5].

```
m = assign(m,'chi',0.9,'xip',100);
% m = assign(m,'chi=',0.9,'xip=',100); [5]
```

Create a database with the new values, and call the function assign.

```
P = struct();
P.chi = 0.9;
P.xip = 100;
m = assign(m,P);
```

Reset the parameters to their original values.

```
m.chi = 0.8;
m.xip = 80;
```

6 Speedy Way to Repeatedly Change Parameters

If you need to iterate over a number of different parameterisations, use the fast version of the function assign. First, initialise the fast assign by specifying the list of parameters (and nothing else) 6. Then, use assign repeatedly to pass different sets of values (in the same order) to the model object 7. Compare the time needed to assign 1,000 different pairs of values for two parameters.

```
load read_model m;

chis = linspace(0.5,0.95,1000);
xips = linspace(60,200,1000);

assign(m,{’chi’,’xip’}); 6

tic
for i = 1 : 1000
   m = assign(m,[chis(i),xips(i)]); 7
end
toc

tic
for i = 1 : 1000
   m.chi = chis(i);
   m.xip = xips(i);
end
toc
```

```
Elapsed time is 0.435049 seconds.
Elapsed time is 1.146393 seconds.
```

7 Assign or Change Steady State Manually

If you wish to manually change some of the steady-state values (or, for instance, assign all of them because they have been computed outside the model), treat the steady-state values the same way as parameters.

```
m = sstate(m,'growth=',true,'blocks=',true,'display=','off');
chksstate(m)
disp('Steady-state database')
sstate_database = get(m,'sstate')
```

```
ans =
     1
Steady-state database
sstate_database =
      Short: 7.1827
       Infl: 2.5000
     Growth: 3.0000
       Wage: 5.5750
          Y: 1.5519 + 1.0074i
          N: 0.7470 + 1.0000i
          W: 1.7314 + 1.0137i
          Q: 0.8333 + 1.0062i
          H: 1.5519 + 1.0074i
          A: 1.0000 + 1.0074i
          P: 1.0000 + 1.0062i
          R: 1.0175 + 1.0000i
         Pk: 1.5312 + 1.0137i
         Rk: 0.0517 + 1.0137i
     Lambda: 0.6444 + 0.9865i
         dP: 1.0062 + 1.0000i
        d4P: 1.0250 + 1.0000i
         dW: 1.0137 + 1.0000i
        RMC: 0.8333 + 1.0000i
         Mp: 0
         Mw: 0
         Ey: 0
         Ep: 0
         Ea: 0
         Er: 0
         Ew: 0
      alpha: 1.0074
       beta: 0.9962
      gamma: 0.6000
      delta: 0.0300
```

```
         k: 10
        pi: 1.0062
       eta: 6
       psi: 0.2500
       chi: 0.9500
       xiw: 60
       xip: 200
      rhoa: 0.9000
      rhor: 0.8500
    kappap: 3.5000
    kappan: 0
    Short_: 0
     Infl_: 0
   Growth_: 0
     Wage_: 0
```

Change both the levels and growth rates of `Y` and `C` using the model-dot-name notation.

```
m.Y = 2 + 1.01i;
m.Pk = 10 + 1.05i;
```

Change the steady states for `Y` and `C` using the function `assign` with name-pair values.

```
m = assign(m,'Y',2+1.01i,'Pk',10+1.05i);
```

Do the same as above but separately for the levels and growth rates.

```
m = assign(m,'-level','Y',2,'Pk',10);
m = assign(m,'-growth','Y',1.01,'Pk',1.05);
```

Change the steady states by creating a database with the new values, and passing the database in `assign`.

```
P = struct();
P.Y = 2 + 1.01i;
P.Pk = 10 + 1.05i;
m = assign(m,P);
```

Note that the newly assigned steady states are, of course, not consistent with the model.

```
disp('Check steady state -- it does not hold');
[flag,list] = chksstate(m,'error=',false);
flag
list.'
```

```
Check steady state -- it does not hold
flag =
     0
ans =
    'Growth=100*((Y/Y{-1})^4-1);'
    'P*Lambda=#(1-chi)/(Y-chi*H)!!P*Y*Lambda=1;'
    'H=exp(Ey)*alpha*Y{-1}!!H=Y;'
    'Lambda*Pk=beta*Lambda{1}*(Rk{1}+(1-delta)*Pk{1});'
    'Y=A*(N-(1-gamma)*&N)^gamma*k^(1-gamma);'
    'gamma*Q*Y=#W*(N-(1-gamma)*&N);'
    '(1-gamma)*Q*Y=Rk*k;'
```

Reset the steady state to the original values.

```
m = assign(m,sstate_database);
disp('Check steady state -- it holds');
chksstate(m)
```

```
Check steady state -- it holds
ans =
     1
```

8 Help on IRIS Functions Used in This File

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

```
help model/model
help model/subsasgn
help model/assign
help model/chksstate
```