∫ IRIS Macroeconomic Modeling Tutorials

SIMPLE SPBC MODEL: GET INFORMATION ABOUT MODEL OBJECT

by Jaromir Benes

January 18, 2016

Summary

Use the function get (and a few others) to access various pieces of information about the model and its properties, such as variable names, parameter values, equations, lag structure, or the model eigenvalues. Two related topics are furthermore covered in separate files: assigning/changing parameters and steady-state values, and accessing model solution matrices.

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>
```

2 LOAD SOLVED MODEL OBJECT AND HISTORICAL DATABASE

Load the solved model object built `read_model`. You must run `read_model` at least once before running this m-file.

```
load read_model.mat m;
```

3 NAMES OF VARIABLES, SHOCKS AND PARAMETERS

```
disp('List of transition variables');
get(m,'xList')

disp('List of measurement variables');
get(m,'yList')

disp('List of shocks');
get(m,'eList')

disp('List of parameters');
get(m,'pList')
```

```
List of transition variables
ans =
  Columns 1 through 10
    'Y'    'N'    'W'    'Q'    'H'    'A'    'P'    'R'    'Pk'    'Rk'
  Columns 11 through 15
    'Lambda'    'dP'    'd4P'    'dW'    'RMC'
List of measurement variables
ans =
    'Short'    'Infl'    'Growth'    'Wage'
```

```
List of shocks
ans =
    'Mp'    'Mw'    'Ey'    'Ep'    'Ea'    'Er'    'Ew'
List of parameters
ans =
  Columns 1 through 8
    'alpha'    'beta'    'gamma'    'delta'    'k'    'pi'    'eta'    'psi'
  Columns 9 through 15
    'chi'    'xiw'    'xip'    'rhoa'    'rhor'    'kappap'    'kappan'
  Columns 16 through 19
    'Short_'    'Infl_'    'Growth_'    'Wage_'
```

4 DESCRIPTION OF VARIABLES, SHOCKS AND PARAMETERS

```
disp('Database with descriptions of all variables, shocks and parameters');
get(m,'descript')

disp('List of descriptions of transition variables');
get(m,'xDescript')
```

```
Database with descriptions of all variables, shocks and parameters
ans =
      Short: 'Short Term Rate'
       Infl: 'Price Inflation'
     Growth: 'Output Growth'
       Wage: 'Wage Inflation'
          Y: 'Output'
          N: 'Labor'
          W: 'Wage rate'
          Q: 'Nominal Marginal Cost'
          H: 'Consumption Habit'
          A: 'Productivity'
          P: 'Final Prices'
          R: 'Interest Rate'
         Pk: 'Price of Capital'
         Rk: 'Rental Price of Capital'
     Lambda: 'Households Shadow Value of Wealth'
         dP: 'Inflation Q/Q'
        d4P: 'Inflation Y/Y'
         dW: 'Wage Inflation Q/Q'
        RMC: 'Real Marginal Cost'
         Mp: 'Measurement Error on Price Inflation'
         Mw: 'Measurement Error on Wage Inflation'
```

```
         Ey: 'Consumption Demand Shock'
         Ep: 'Cost Push Shock'
         Ea: 'Productivity Shock'
         Er: 'Policy Shock'
         Ew: 'Wage Shock'
      alpha: 'Long Run Growth'
       beta: 'Discount'
      gamma: 'Labor Share'
      delta: 'Depreciation'
          k: ''
         pi: ''
        eta: ''
        psi: ''
        chi: 'Habit'
        xiw: 'Wage Stickiness'
        xip: 'Price Stickiness'
       rhoa: ''
       rhor: ''
     kappap: ''
     kappan: ''
     Short_: ''
      Infl_: ''
    Growth_: ''
      Wage_: ''
List of descriptions of transition variables
ans =
  Columns 1 through 4
    'Output'    'Labor'    'Wage rate'    'Nominal Marginal...'
  Columns 5 through 8
    'Consumption Habit'    'Productivity'    'Final Prices'    'Interest Rate'
  Columns 9 through 11
    'Price of Capital'    'Rental Price of ...'      'Households Shado...'
  Columns 12 through 14
    'Inflation Q/Q'    'Inflation Y/Y'    'Wage Inflation Q/Q'
  Column 15
    'Real Marginal Cost'
```

5 EQUATIONS AND EQUATION LABELS

```
disp('Transition equations')
xeqtn = get(m,'xEqtns');
xeqtn'

disp('Measurement equations')
```

```
yeqtn = get(m,'yEqtn');
yeqtn'

disp('Transition equation labels')
get(m,'xLabels')

disp('Equation with whose label is Production function');
findeqtn(m,'Production function')

disp('Equations whose labels start with P');
eqtn = findeqtn(m,'-rexp','P.*');
eqtn{:}
```

```
Transition equations
ans =
    'P*Lambda=#(1-chi)/(Y-chi*H)!!P*Y*Lambda=1;'
    'Lambda=beta*R*Lambda{1}!!beta*R=alpha*pi;'
    'H=exp(Ey)*alpha*Y{-1}!!H=Y;'
    'xiw/(eta-1)*(dW/dW{-1}-1)=beta*xiw/(eta-1)*(dW{1}/dW-1+Ew)+(eta/(eta-1...'
    '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;'
    'xip/(eta-1)*(dP/dP{-1}-1)=beta*xip/(eta-1)*(dP{1}/dP-1+Ep)+(eta/(eta-1...'
    'RMC=Q/P!!RMC=(eta-1)/eta;'
    'log(A/A{-1})=rhoa*log(A{-1}/A{-2})+(1-rhoa)*log(alpha)+Ea;'
    'log(R)=rhor*log(R{-1})+(1-rhor)*(log(&R)+kappap*(log(dP{4})-log(pi))+k...'
    'dP=P/P{-1};'
    'd4P=P/P{-4};'
    'dW=W/W{-1};'
Measurement equations
ans =
    'Short=100*(R^4-1);'
    'Infl=100*((P/P{-1})^4-1+Mp);'
    'Wage=100*((W/W{-1})^4-1+Mw);'
    'Growth=100*((Y/Y{-1})^4-1);'
Transition equation labels
ans =
  Columns 1 through 6
    ''    ''    ''    'Wage Phillips Curve'    ''    'Production Function'
  Columns 7 through 15
    ''    ''    'Price Phillips C...'      ''    ''    ''    ''    ''    ''
Equation with whose label is Production function
ans =
     []
Equations whose labels start with P
```

```
ans =
Y=A*(N-(1-gamma)*&N)^gamma*k^(1-gamma);
ans =
xip/(eta-1)*(dP/dP{-1}-1)=beta*xip/(eta-1)*(dP{1}/dP-1+Ep)+(eta/(eta-1)*RMC-1)!!eta/(eta-1)*Q=P;
```

6 COMMENTS AND USER DATA

Assign a text comment or any kind of user data to a model object using the functions comment and userdata, respectively. The same functions are also used to get the current comment or the user data. It's only your business whether and how you use these.

```
c = comment(m)

m = comment(m,'New comment');
comment(m)

m = comment(m,c);

x = struct();
x.ToDo = 'Fix this and that';
x.SomeRandNumbers = rand(1,10);

m = userdata(m,x)

userdata(m)
```

```
c =
Simple Sticky Price Business Cycle Model File
ans =
New comment
m =
        nonlinear model object: [1] parameterisation(s)
        number of equations: [4 15 4 0 0]
        solution(s) available: [1] parameterisation(s)
        comment: 'Simple Sticky Price Business Cycle Model File'
        user data:
                ToDo: 'Fix this and that'
                SomeRandNumbers: [0.547095959676547 0.541268467485969 0.788113184048992 0.869605555315031 0.78754053
        export files: [0]

ans =
               ToDo: 'Fix this and that'
    SomeRandNumbers: [1x10 double]
```

7 DIFFERENT WAYS TO GET AND ASSIGN/CHANGE PARAMETERS

There are multiple equivalent ways how to view and assign parameters. Display the parameter 'gamma', and change the values for two std deviations, 'std_ep' and 'std_ew'.

```
P = get(m,'parameters');
P.gamma

m.gamma

s = struct();
s.std_Ey = 0.02;
s.std_Ep = 0.02;
m = assign(m,s);

m = assign(m,'std_Ey',0.02,'std_Ep',0.02);

m.std_Ey = 0.02;
m.std_Ep = 0.02;
```

```
ans =
    0.6000
ans =
    0.6000
```

8 CHECK STATIONARITY

The logical value `true` is displayed as `1`, the logical value `false` is displayed as `0`.

```
disp('Is the model stationary?');
isstationary(m)

disp('Is the variable stationary?');
get(m,'stationary')

disp('List of stationary variables');
get(m,'stationaryList')

disp('List of non-stationary variables');
get(m,'nonstationaryList')
```

```
Is the model stationary?
```

```
ans =
     0
Is the variable stationary?
ans =
     Short: 1
      Infl: 1
    Growth: 1
      Wage: 1
         N: 1
         Q: 0
         H: 0
        Pk: 0
        Rk: 0
    Lambda: 0
       d4P: 1
       RMC: 1
         Y: 0
         W: 0
         A: 0
         P: 0
         R: 1
        dP: 1
        dW: 1
List of stationary variables
ans =
  Columns 1 through 8
    'Short'    'Infl'    'Growth'    'Wage'    'N'    'd4P'    'RMC'    'R'
  Columns 9 through 10
    'dP'    'dW'
List of non-stationary variables
ans =
    'Q'    'H'    'Pk'    'Rk'    'Lambda'    'Y'    'W'    'A'    'P'
```

9 GET CURRENTLY ASSIGNED STEADY STATE

Steady state is described by complex numbers:

- real part = steady-state levels
- imaginary part = steady-state growth

The interpretation of the steady-state growth rates differs for linearised versus log-linearised variables: * linearised variables: x(t) - x(t-1) * log-linearised variables: x(t) / x(t-1)

```
disp('Steady-state levels and growth rates');
get(m,'sstate')

disp('Steady-state levels');
get(m,'sstateLevel')

disp('Steady-state growth rates')
get(m,'sstateGrowth')

disp('Is the variable a log-variable?');
get(m,'log')

disp('List of log-variables');
get(m,'logList')
```

```
Steady-state levels and growth rates
ans =
      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: 0.8500
        xiw: 60
        xip: 300
       rhoa: 0.9000
       rhor: 0.8500
     kappap: 3.5000
     kappan: 0
     Short_: 0
      Infl_: 0
    Growth_: 0
      Wage_: 0
Steady-state levels
ans =
      Short: 7.1827
       Infl: 2.5000
     Growth: 3.0000
       Wage: 5.5750
          Y: 1.5519
          N: 0.7470
          W: 1.7314
          Q: 0.8333
          H: 1.5519
          A: 1.0000
          P: 1
          R: 1.0175
         Pk: 1.5312
         Rk: 0.0517
     Lambda: 0.6444
         dP: 1.0062
        d4P: 1.0250
         dW: 1.0137
        RMC: 0.8333
         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: 0.8500
        xiw: 60
        xip: 300
       rhoa: 0.9000
       rhor: 0.8500
     kappap: 3.5000
     kappan: 0
     Short_: 0
      Infl_: 0
    Growth_: 0
      Wage_: 0
Steady-state growth rates
ans =
      Short: 0
       Infl: 0
     Growth: 0
       Wage: 0
          Y: 1.0074
          N: 1
          W: 1.0137
          Q: 1.0062
          H: 1.0074
          A: 1.0074
          P: 1.0062
          R: 1
         Pk: 1.0137
         Rk: 1.0137
     Lambda: 0.9865
         dP: 1
        d4P: 1
         dW: 1
        RMC: 1
         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: 0.8500
        xiw: 60
        xip: 300
       rhoa: 0.9000
       rhor: 0.8500
     kappap: 3.5000
     kappan: 0
     Short_: 0
      Infl_: 0
    Growth_: 0
      Wage_: 0
Is the variable a log-variable?
ans =
     Short: 0
      Infl: 0
    Growth: 0
      Wage: 0
         Y: 1
         N: 1
         W: 1
         Q: 1
         H: 1
         A: 1
         P: 1
         R: 1
        Pk: 1
        Rk: 1
    Lambda: 1
        dP: 1
       d4P: 1
        dW: 1
       RMC: 1
        Mp: 0
        Mw: 0
        Ey: 0
        Ep: 0
        Ea: 0
        Er: 0
```

```
       Ew: 0
List of log-variables
ans =
  Columns 1 through 10
    'Y'    'N'    'W'    'Q'    'H'    'A'    'P'    'R'    'Pk'    'Rk'
  Columns 11 through 15
    'Lambda'    'dP'    'd4P'    'dW'    'RMC'
```

10 LAGS AND INITIAL CONDITIONS

```
disp('Maximum lag in the model');
get(m,'maxLag')

disp('List of initial conditions needed for simulations and forecasts');
get(m,'required')
```

```
Maximum lag in the model
ans =
    -4
List of initial conditions needed for simulations and forecasts
ans =
  Columns 1 through 4
    'log(Y{-1})'    'log(W{-1})'    'log(A{-1})'    'log(P{-1})'
  Columns 5 through 8
    'log(R{-1})'    'log(dP{-1})'    'log(dW{-1})'    'log(A{-2})'
  Columns 9 through 11
    'log(P{-2})'    'log(P{-3})'    'log(P{-4})'
```

11 EIGENVALUES

Get stable, unit, or unstable eigenvalues (roots). Plot the stable roots in a unit circle. Display the dominant (largest) stable root, and the dominant (smallest) unstable root.

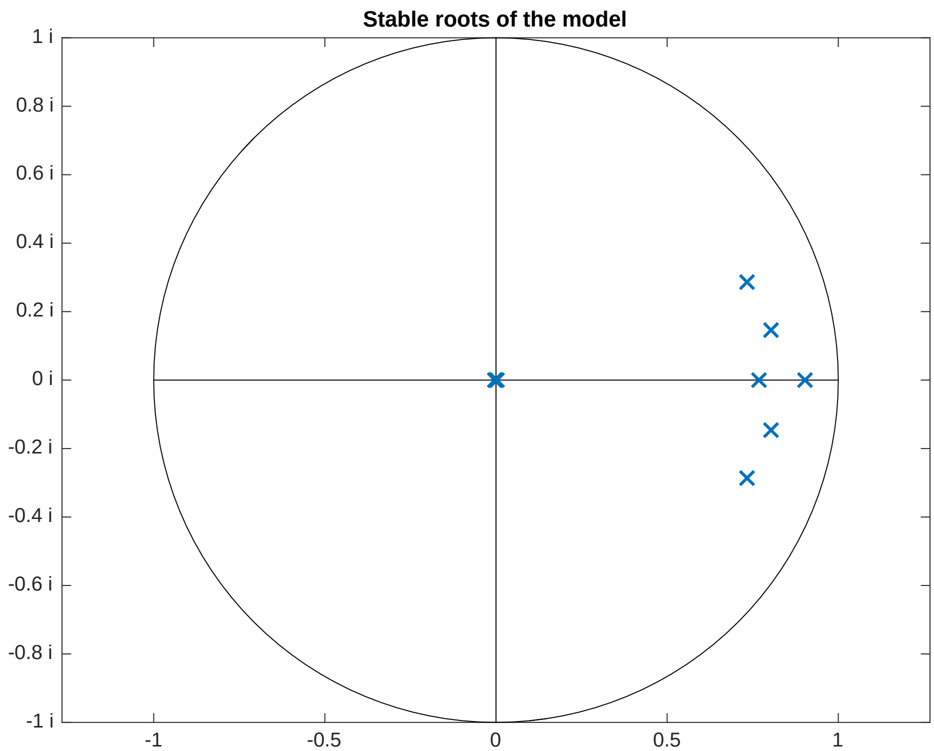
```
format('short','e');

disp('Model eigenvalues');
all_roots = get(m,'roots');
all_roots.'

stable_roots = get(m,'stableRoots');
unit_roots = get(m,'unitRoots');
```

```
unstable_roots = get(m,'unstableRoots');

disp('Stable roots');
stable_roots.'

disp('Unit roots');
unit_roots.'

disp('Unstable roots');
unstable_roots.'

format();

figure();
ploteig(stable_roots);
title('Stable roots of the model');

[~,index] = sort(abs(stable_roots),'descend');
stable_roots = stable_roots(index);
[~,index] = sort(abs(unstable_roots),'ascend');
unstable_roots = unstable_roots(index);

disp('Largest stable root');
stable_roots(1)
disp('Smallest unstable root and its inverse');
[unstable_roots(1),1./unstable_roots(1)]
```

```
Model eigenvalues
ans =
   1.0000e+00 + 0.0000e+00i
   1.0000e+00 + 0.0000e+00i
  -1.6530e-17 + 0.0000e+00i
   4.0047e-18 + 0.0000e+00i
   7.3579e-01 + 2.8346e-01i
   7.3579e-01 - 2.8346e-01i
   7.6951e-01 + 0.0000e+00i
   8.0212e-01 + 1.4862e-01i
   8.0212e-01 - 1.4862e-01i
  -4.4181e-18 + 0.0000e+00i
  -1.4380e-20 + 0.0000e+00i
  -4.7708e-18 + 0.0000e+00i
   9.0000e-01 + 0.0000e+00i
   1.0348e+00 + 0.0000e+00i
   6.8154e+16 + 0.0000e+00i
   7.9528e+16 + 0.0000e+00i
  -1.7709e+16 + 0.0000e+00i
```

```
  -1.9247e+00 + 3.3797e+01i
  -1.9247e+00 - 3.3797e+01i
   1.0980e+00 + 2.8748e-01i
   1.0980e+00 - 2.8748e-01i
   1.7634e+00 + 0.0000e+00i
  -4.2760e+14 + 0.0000e+00i
   2.4704e+14 + 0.0000e+00i
   2.7189e+15 + 0.0000e+00i
   1.2271e+16 + 0.0000e+00i
Stable roots
ans =
  -1.6530e-17 + 0.0000e+00i
   4.0047e-18 + 0.0000e+00i
   7.3579e-01 + 2.8346e-01i
   7.3579e-01 - 2.8346e-01i
   7.6951e-01 + 0.0000e+00i
   8.0212e-01 + 1.4862e-01i
   8.0212e-01 - 1.4862e-01i
  -4.4181e-18 + 0.0000e+00i
  -1.4380e-20 + 0.0000e+00i
  -4.7708e-18 + 0.0000e+00i
   9.0000e-01 + 0.0000e+00i
Unit roots
ans =
   1.0000e+00
   1.0000e+00
Unstable roots
ans =
   1.0348e+00 + 0.0000e+00i
   6.8154e+16 + 0.0000e+00i
   7.9528e+16 + 0.0000e+00i
  -1.7709e+16 + 0.0000e+00i
  -1.9247e+00 + 3.3797e+01i
  -1.9247e+00 - 3.3797e+01i
   1.0980e+00 + 2.8748e-01i
   1.0980e+00 - 2.8748e-01i
   1.7634e+00 + 0.0000e+00i
  -4.2760e+14 + 0.0000e+00i
   2.4704e+14 + 0.0000e+00i
   2.7189e+15 + 0.0000e+00i
   1.2271e+16 + 0.0000e+00i
Largest stable root

ans =

    0.9000
```

```
Smallest unstable root and its inverse

ans =

    1.0348    0.9663
```



12 HELP ON IRIS FUNCTIONS USED IN THIS FILE

```
help model/comment
help model/findeqtn
help model/get
help model/subsref
help model/subsasgn
help model/userdata
```