Simulate Simple Shock Responses

simulate_simple_shock.m

by Jaromir Benes

May 27, 2015

Summary

Simulate a simple shock both as deviations from control and in full levels, and report the simulation results.

Contents

1 Clear Workspace

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

```
clear;
close all;
clc;
irisrequired 20140315;
```

2 Load Solved Model Object

Load the solved model object built in `read_model`. Run `read_model` at least once before running this m-file.

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

3 Define Dates

Define the start and end dates as plain numbered periods here.

```
startDate = 1;
endDate = 40;
```

Alternatively, use the IRIS functions `yy`, `hh`, `qq`, `bb`, or `mm` to create and use proper dates (with yearly, half-yearly, quarterly, bi-monthly, or monthly frequency, respectively).

```
startdate = qq(2010,1);
enddate = startdate + 39;
```

4 Simulate Consumption Demand Shock

Simulate the shock as deviations from control (e.g. from the steady state or balanced-growth path). To this end, set the option `'deviation='` to true. Both the input and output database are then interpreted as deviations from control:

- the deviations for linearised variables are defined as $x_t - x_t$: hence, 0 means the variable is on its steady state.

- the deviations for log-linearised variables are defined as $x_t/\bar{x}_t$: hence, 1 means the variable is on its steady state, or 1.05 means it is 5 % above it.

The function `zerodb` automatically detects the maximum lag in the model, and creates the input database accordingly so that it includes all necessary initial conditions.

```
57 d = zerodb(m,startDate:endDate);
58 d.Ey(startDate) = log(1.01);
59 s = simulate(m,d,1:40,'deviation=',true);
60 s = dboverlay(d,s);
61 s %#ok<NOPTS>
62
63 s1 = simulate(m,d,1:40,'deviation=',true);
```

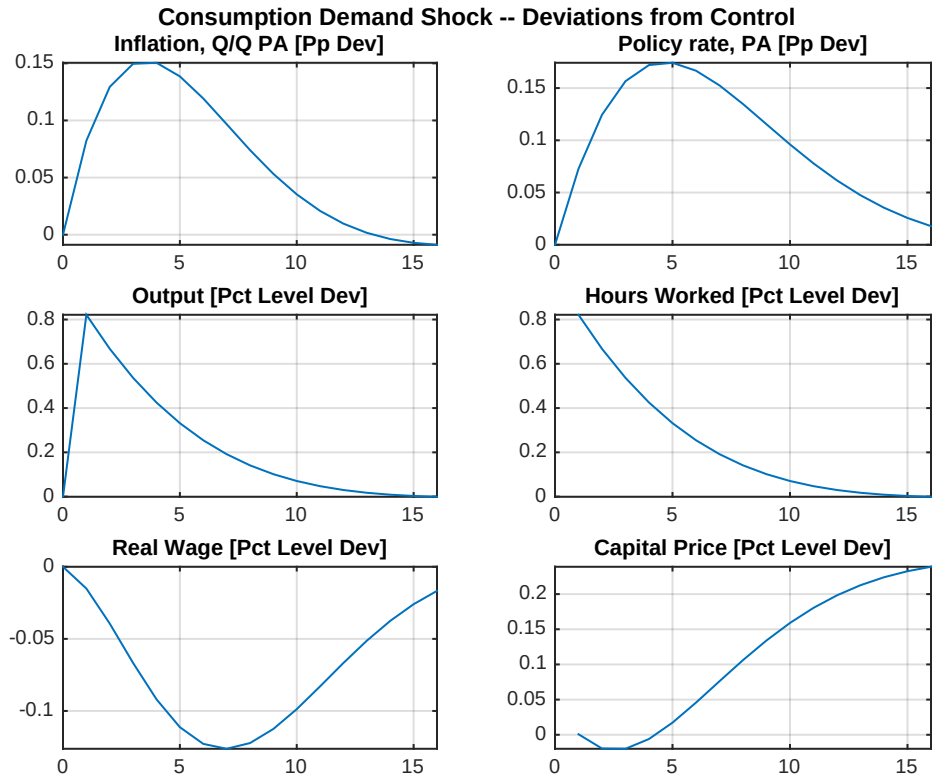
```
s =
      Short: [44x1 tseries]
       Infl: [44x1 tseries]
     Growth: [44x1 tseries]
       Wage: [44x1 tseries]
          Y: [44x1 tseries]
          N: [44x1 tseries]
          W: [44x1 tseries]
          Q: [44x1 tseries]
          H: [44x1 tseries]
          A: [44x1 tseries]
          P: [44x1 tseries]
          R: [44x1 tseries]
         Pk: [44x1 tseries]
         Rk: [44x1 tseries]
     Lambda: [44x1 tseries]
         dP: [44x1 tseries]
        d4P: [44x1 tseries]
         dW: [44x1 tseries]
        RMC: [44x1 tseries]
         Mp: [40x1 tseries]
         Mw: [40x1 tseries]
         Ey: [40x1 tseries]
         Ep: [40x1 tseries]
         Ea: [40x1 tseries]
         Er: [40x1 tseries]
         Ew: [40x1 tseries]
      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
     ttrend: [44x1 tseries]
```

5 Report Simulation Results

Use the dbplot function to create a quick report of simulation results. Note how we use the 'transform=' option [1] to plot percent deviations of individual variables.

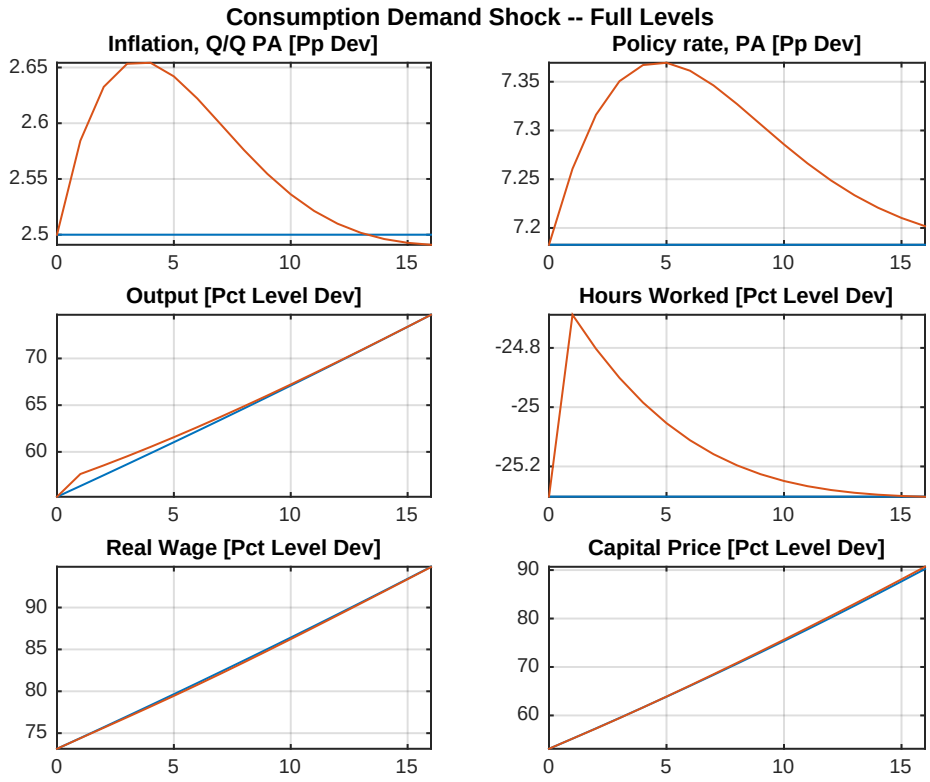
```
plotRng = startDate-1 : startDate+15;
plotList = { ...
    ' "Inflation, Q/Q PA [Pp Dev]" dP^4 ', ...
        ' "Policy rate, PA [Pp Dev]" R^4 ', ...
    ' "Output [Pct Level Dev]" Y ', ...
    ' "Hours Worked [Pct Level Dev]" N ', ...
    ' "Real Wage [Pct Level Dev]" W/P ', ...
    ' "Capital Price [Pct Level Dev]" Pk', ...
   };
dbplot(s1,plotRng,plotList, ...
   'tight=',true,'transform=',@(x) 100*(x-1)); [1]
grfun.ftitle('Consumption Demand Shock -- Deviations from Control');
```



6 Simulate Shock in Full Levels

Instead of deviations from control, simulate now the same shocks in full levels. To that end, create an input dabase with the steady state (balanced-growth path) using `sstatedb`, and keep the option `'deviation='` false (default). When reporting the results, plot both the simulated shock against the steady-state (balanced-growth path) database: The `&` operator [2] combines two databases so that every time series has two columns.

```
94  d = sstatedb(m,startDate:endDate);
95  d.Ey(startDate) = log(1.01);
96  s = simulate(m,d,1:40);
97  s = dboverlay(d,s);
98
99  dbplot(d & s,plotRng,plotList, ... 2
100    'tight=',true,'transform=',@(x) 100*(x-1));
101 grfun.ftitle('Consumption Demand Shock -- Full Levels');
```



7 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.

```
help model/simulate
help model/sstatedb
help model/zerodb
help dbase/dbplot
help grfun/ftitle
help dbase/dboverlay
```