More Complex Simulation Experiments

simulate_complex_shocks.m

by Jaromir Benes

May 27, 2015

Summary

Simulate the differences between anticipated and unanticipated future shocks, run experiments with temporarily exogenised variables, and show how easy it is to examine simulations with mutliple different parameterisations.

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.

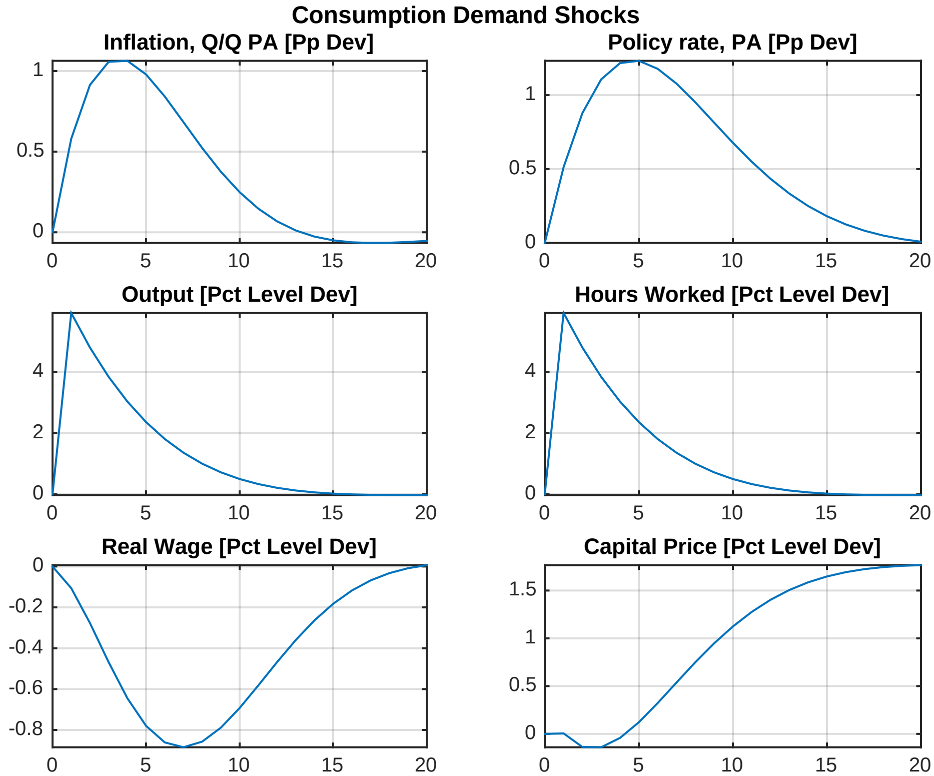
```
load read_model.mat m;
```

3 Define Dates and Ranges

```
startDate = 1;
endDate = 40;
plotRng = startDate-1 : startDate+19;
```

4 Simple Consumption Demand Shock

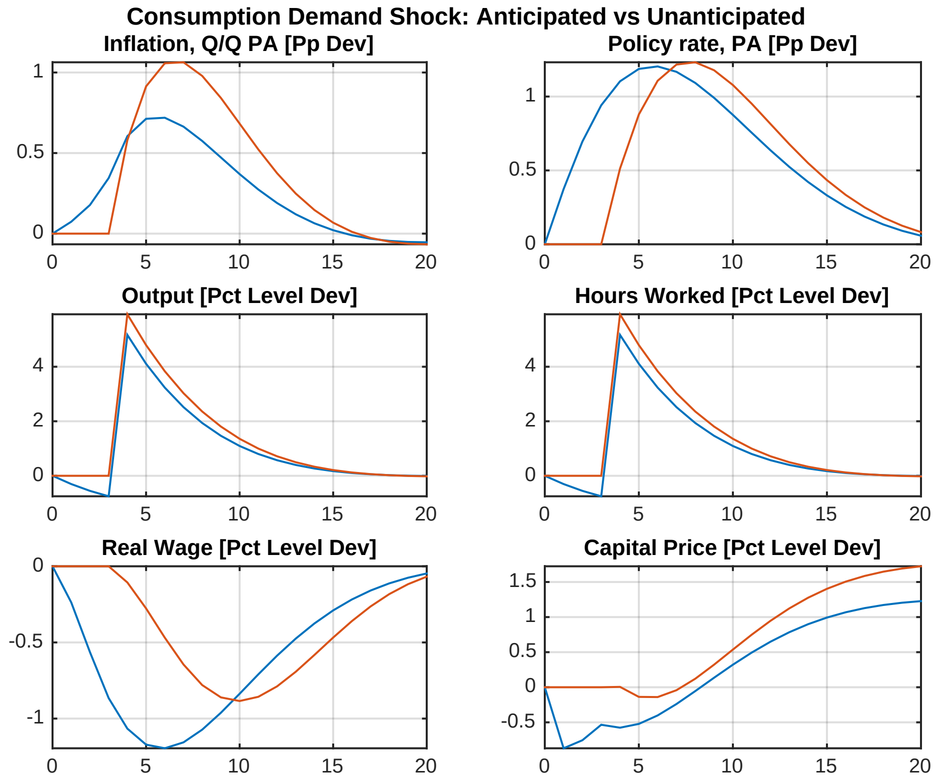
```
d = zerodb(m,startDate-3:startDate);
d.Ey(startDate) = 0.07;
s = simulate(m,d,1:40,’deviation’,true);
s = dbextend(d,s);

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(s,plotRng,plotList,’tight=’,true,’transform=’,@(x) 100*(x-1));
grfun.ftitle(’Consumption Demand Shocks’);
```



5 Anticipated vs Unanticipated Consumption Demand Shock

Simulate a future (3 quarters ahead) aggregate demand shock twice: as anticipated and unanticipated.

```
d = zerodb(m,startDate-3:startDate);
d.Ey(startDate+3) = 0.07;
s1 = simulate(m,d,startDate:endDate,'deviation=',true,'anticipate=',true);
s1 = dbextend(d,s1);

s2 = simulate(m,d,startDate:endDate,'deviation=',true,'anticipate=',false);
s2 = dbextend(d,s2);

dbplot(s1 & s2,plotRng,plotList, ...
    'tight=',true,'transform=',@(x) 100*(x-1));
grfun.ftitle('Consumption Demand Shock: Anticipated vs Unanticipated');
grfun.bottomlegend('Anticipated','Unanticipated');
```



6 Simulate Consumption Demand Shock with Delayed Policy Reaction

Simulate a consumption shock and, at the same time, delay the policy reaction (by exogenising the policy rate to its pre-shock level for 3 periods). Again, this can be done in an anticipated mode and unanticipated mode.

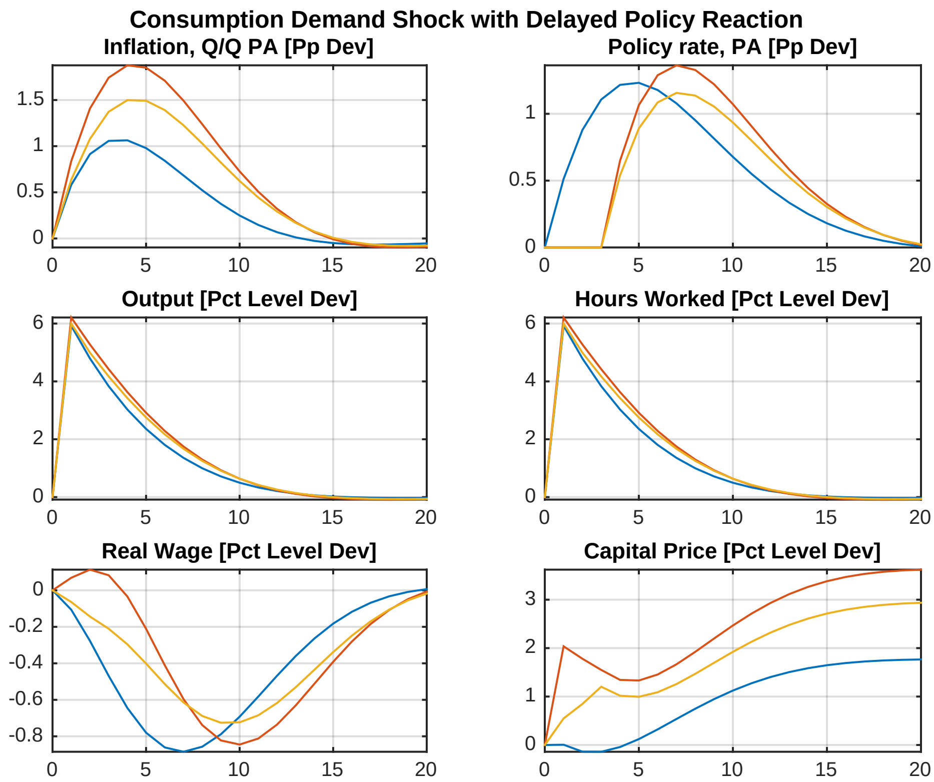
1 Simulates consumption shocks with immediate policy reaction.

2 Simulates the same shock with delayed policy reaction that is announced and anticipated from the beginning.

3 Simulates the same shock with delayed policy reaction that takes everyone by surprise every quarter.

```
nPer = 3;

d = zerodb(m,startDate-3:startDate);
```

```
d.Ey(startDate) = 0.07;

p = plan(m,startDate:endDate);
p = exogenise(p,'R',startDate:startDate+nPer-1);
p = endogenise(p,'Er',startDate:startDate+nPer-1);
d.R(startDate:startDate+nPer-1) = 1;

s1 = simulate(m,d,startDate:endDate, ... 1
   'deviation',true);
s1 = dbextend(d,s1);

s2 = simulate(m,d,startDate:endDate, ... 2
   'deviation',true,'plan',p);
s2 = dbextend(d,s2);

s3 = simulate(m,d,startDate:endDate, ... 3
   'deviation',true,'plan',p,'anticipate',false);
s3 = dbextend(d,s3);

dbplot(s1 & s2 & s3,plotRng,plotList, ...
    'tight=',true,'transform=',@(x) 100*(x-1));
grfun.ftitle('Consumption Demand Shock with Delayed Policy Reaction');
grfun.bottomlegend('No delay','Anticipated','Unanticipated');
```



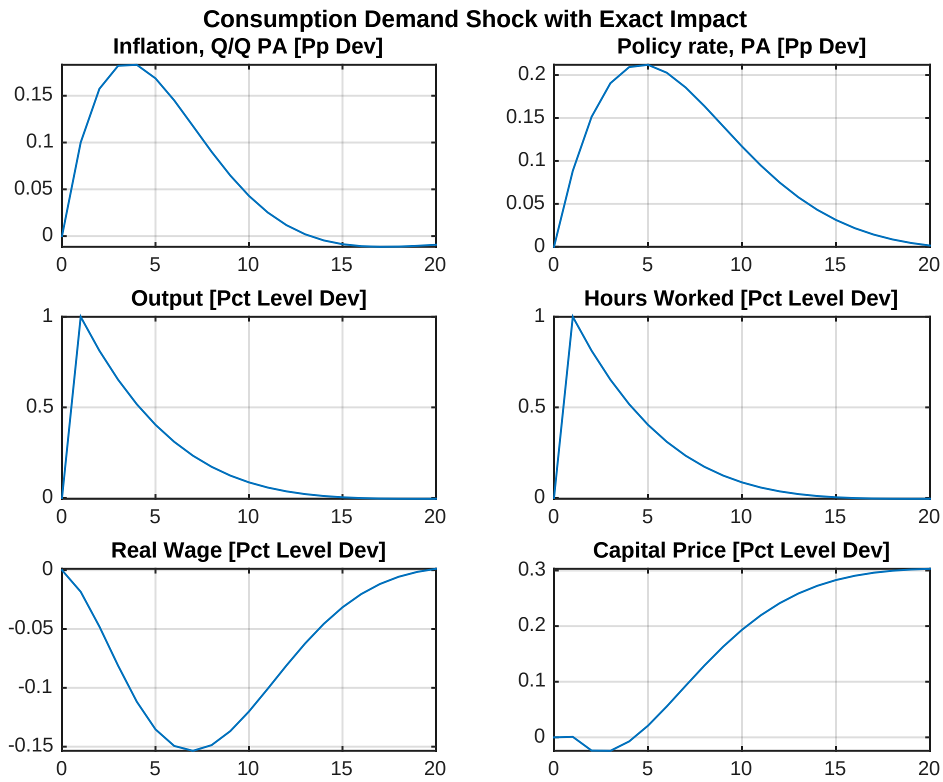
7 Simulate Consumption Demand Shock with Desired Impact

Find the size of a consumption demand shock such that it leads to exactly a 1 pct increase in consumption in the first period. Because consumption (C) is a log-linearised variable, specify the 1 pct deviation from its steady state as 1.01.

```
d = zerodb(m,startDate-3:startDate);
d.Y(startDate) = 1.01;

p = plan(m,startDate:endDate);
p = exogenise(p,'Y',startDate);
p = endogenise(p,'Ey',startDate);
s = simulate(m,d,startDate:endDate,'deviation=',true,'plan=',p);
s = dbextend(d,s);

disp(s.Ey{1:10});

dbplot(s,plotRng,plotList,'tight=',true,'transform=',@(x) 100*(x-1));
```

```
grfun.ftitle('Consumption Demand Shock with Exact Impact');
```

```
        tseries object: 10-by-1
         1:  0.012103
         2:         0
         3:         0
         4:         0
         5:         0
         6:         0
         7:         0
         8:         0
         9:         0
        10:         0
        'Consumption Demand Shock'
        user data: empty
        export files: [0]
```

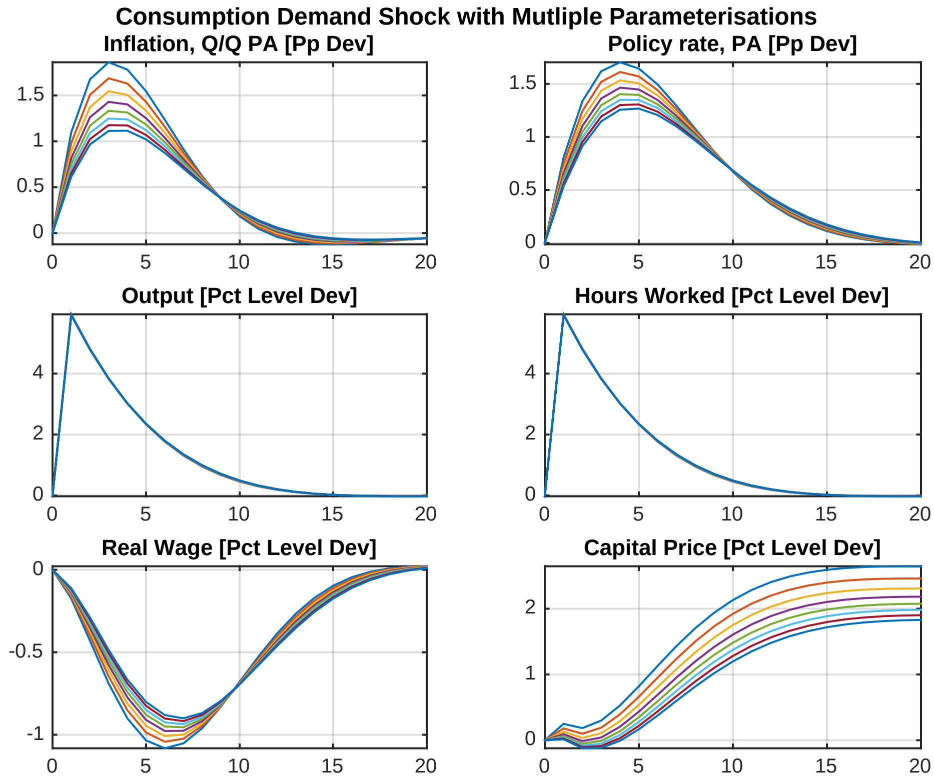


8 Simulate Consumption Demand Shocks with Multiple Parameterisations

Within the same model object, expand the number of its parameterisations, and assign different sets of values to some (or all) of the parameters (here, only the values for xi vary, i.e. the price adjustment costs). Solve and simulate all these different parameterisations at once. Note that virtually all IRIS functions support multiple parameterisations.

```
m(1:8) = m;
m.xip = [140,160,180,200,220,240,260,280];
m = solve(m);
disp(m);

d = zerodb(m,startDate-3:startDate);
d.Ey(1,:) = 0.07;

s = simulate(m,d,startDate:endDate,'deviation=',true);
s = dbextend(d,s);

dbplot(s,plotRng,plotList,'tight=',true,'transform=',@(x) 100*(x-1));
grfun.ftitle('Consumption Demand Shock with Mutliple Parameterisations');
```

```
        nonlinear model object: [8] parameterisation(s)
        number of equations: [4 15 4 0 0]
        solution(s) available: [8] parameterisation(s)
        comment: 'Simple Sticky Price Business Cycle Model'
        user data: empty
        export files: [0]
```



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.

```
help model/dbextend
help model/simulate
help model/solve
help model/subsasgn
help model/zerodb
help qreport/qplot
help grfun/ftitle
help dbextend
```