Simulate Fisher Info Matrix and Test Parameter Identification

fisher_information_matrix.m

by Jaromir Benes

May 27, 2015

Summary

Calculate estimates of the Fisher information matrix. The Fisher matrix is a property of the model itself, and is independent of any data. It represents the maximum amount of information one can hope for to find in the data in case the data are really generated by the model DGP.

Compare two approaches: a time-domain approach, and a frequency-domain approach. Use the singular value decomposition to learn more about which parameters (or combinations of them) are identified the best or the worst.

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 m;
```

3 Calculate Fisher Information Matrix

The Fisher information matrix is a useful tools examining how much information can be recovered from the data to identify some of the model parameters (under the assumption that the model with its current parameters is the true DGP).

Compute the Fisher information matrix using two approaches: * a simulation method in the time domain. * an analytical method in the frequency domain.

The results are asymptotically equivalent. The actual differencies observed in this exercise arise because of

- sampling errors in the time domain,
- the fact that the frequency domain approach assumes that the model variables follow a circular process (which is not true for samples of finite lenthgs).

The differences and other approximation errors are though usually immaterial for detecting identification deficiencies.

```
rng(0);

% List of parameters for which for which the Fisher matrix will be
% evaluated.
```

```
plist = {'chi','xiw','xip','rhor','kappap','kappan','std_Ey'};

% Set to a larger number in practice.
nsim = 100;

fprintf('Resample %g times from calibrated model.\n',nsim);

% Simulate a total of nsim artificial data, length 40 periods.
d = resample(m,[],1:40,nsim,'deviation=',false);
d = rmfield(d,'Wage');

disp('Compute Hessians for each draw and average them.');

[mloglik,s,F1] = diffloglik(m,d,1:40,plist, ...
   'deviation=',true,'relative=',false,'progress=',true);

F1 = mean(F1,3);

disp('Compute information matrix in frequency domain.');
[F2,F2i,d] = fisher(m,40,plist, ...
    'deviation=',false,'exclude=',{'Wage'},'progress=',true);

format shortEng;
disp('Compare time-domain and frequency domain info matrices.');
disp('Time domain');
F1 %#ok<NOPTS>
disp('Frequency domain');
F2 %#ok<NOPTS>

disp('Compare diagonal elements');
[diag(F1),diag(F2)] %#ok<NOPTS>
format();
```

```
Resample 100 times from calibrated model.
Compute Hessians for each draw and average them.
[--IRIS model.diffloglik progress--------]
[****************************************]
Compute information matrix in frequency domain.
[--IRIS model.fisher progress------------]
[****************************************]
Compare time-domain and frequency domain info matrices.
Time domain
F1 =
  Columns 1 through 4
     1.3081e+003   119.4931e-003   -26.2125e-003  -794.2254e+000
   119.4931e-003   116.2883e-006    19.9635e-006  -148.1386e-003
```

```
   -26.2125e-003    19.9635e-006    72.5923e-006  -113.8912e-003
  -794.2254e+000  -148.1386e-003  -113.8912e-003     1.2426e+003
    18.2865e+000     4.3979e-003     3.9111e-003   -32.3868e+000
    57.6701e+000    12.7627e-003    34.4557e-003  -161.1329e+000
    12.6750e+003     1.4659e+000  -715.9566e-003    -1.3262e+003
  Columns 5 through 7
    18.2865e+000    57.6701e+000    12.6750e+003
     4.3979e-003    12.7627e-003     1.4659e+000
     3.9111e-003    34.4557e-003  -715.9566e-003
   -32.3868e+000  -161.1329e+000    -1.3262e+003
     1.0236e+000     4.2663e+000   -21.8780e+000
     4.2663e+000    35.3534e+000    -1.2123e+003
   -21.8780e+000    -1.2123e+003   970.3600e+003
Frequency domain
F2 =
  Columns 1 through 4
     1.0873e+003    77.0004e-003   -14.7501e-003  -701.1530e+000
    77.0004e-003   113.6707e-006    25.7354e-006  -121.8888e-003
   -14.7501e-003    25.7354e-006    42.9864e-006   -61.8098e-003
  -701.1530e+000  -121.8888e-003   -61.8098e-003   966.6718e+000
    16.8336e+000     4.1748e-003     2.6464e-003   -27.3533e+000
    62.3300e+000    13.9524e-003    12.0337e-003  -104.3835e+000
     5.3613e+003    57.5726e-003   -66.9019e-003     1.0088e+003
  Columns 5 through 7
    16.8336e+000    62.3300e+000     5.3613e+003
     4.1748e-003    13.9524e-003    57.5726e-003
     2.6464e-003    12.0337e-003   -66.9019e-003
   -27.3533e+000  -104.3835e+000     1.0088e+003
   927.4078e-003     3.0535e+000   -53.0061e+000
     3.0535e+000    16.8948e+000  -827.5607e+000
   -53.0061e+000  -827.5607e+000   733.7568e+003
Compare diagonal elements
ans =
     1.3081e+003     1.0873e+003
   116.2883e-006   113.6707e-006
    72.5923e-006    42.9864e-006
     1.2426e+003   966.6718e+000
     1.0236e+000   927.4078e-003
    35.3534e+000    16.8948e+000
   970.3600e+003   733.7568e+003
```

4 Singular Value Decomposition

The singular value decomposition is a quick way how to find out which parameters or combinations of parameters are identified the best or the worst.

```
% TODO: Correct for the absolute size of the parameters.

[u1,s1] = svd(F1);
[u2,s2] = svd(F2);

s1 = diag(s1);
s2 = diag(s2);
s1 = s1 / s1(1);
s2 = s2 / s2(1);

format('shortEng');
disp('Singular values (normalised and ordered)');
disp('Time domain');
disp(s1.');
disp('Frequency domain');
disp(s2.');

disp('Combinations of parameters ordered by degree of identification.');
disp('Best identified columns ordered first');

disp('Time domain');
[char(plist),num2str(u1,'| %-.2g')] %#ok<NOPTS>
disp('Frequency domain');
[char(plist),num2str(u2,'| %-.2g')] %#ok<NOPTS>
format();
```

```
Singular values (normalised and ordered)
Time domain
  Columns 1 through 4

     1.0000e+000     2.0455e-003   434.3808e-006    11.2824e-006

  Columns 5 through 7

   175.9514e-009    97.8515e-012    25.6651e-012

Frequency domain
  Columns 1 through 4

     1.0000e+000     2.3515e-003   408.6947e-006     6.4699e-006
```

```
  Columns 5 through 7

   187.1019e-009   132.3000e-012    31.9893e-012

Combinations of parameters ordered by degree of identification.
Best identified columns ordered first
Time domain

ans =

chi   | -0.013  | -0.68   | 0.73    | -0.045  | -0.003  | 3.1e-06 | -6.6e-05
xiw   | -1.5e-06| -8.9e-05| -6.2e-05| 0.0004  | -0.0035 | 0.99    | 0.17
xip   | 7.4e-07 | -3.8e-05| -0.00022| -0.0013 | -0.0046 | 0.17    | -0.99
rhor  | 0.0014  | 0.73    | 0.67    | -0.16   | -0.029  | 3.8e-05 | 0.00017
kappap| 2.2e-05 | -0.018  | -0.02   | 0.013   | -1      | -0.0042 | 0.0039
kappan| 0.0012  | -0.086  | -0.14   | -0.99   | -0.0081 | 0.00012 | 0.0014
std_Ey| -1      | 0.0098  | -0.0088 | -0.00086| -3.3e-05| -1.3e-06| 1.9e-06

Frequency domain

ans =

chi   | -0.0073 | -0.72   | 0.69    | -0.015   | -0.0052| 1.2e-05 | -7.8e-05
xiw   | -7.9e-08| -8.1e-05| -0.00012| -0.00015 | -0.0048| 0.98    | 0.18
xip   | 9.1e-08 | -1.9e-05| -0.00018| -0.00093 | -0.004 | 0.18    | -0.98
rhor  | -0.0014 | 0.69    | 0.72    | -0.12    | -0.031 | -4e-05  | 8.2e-05
kappap| 7.2e-05 | -0.018  | -0.027  | -0.0039  | -1     | -0.0054 | 0.003
kappan| 0.0011  | -0.07   | -0.095  | -0.99    | 0.0077 | -0.00029| 0.00087
std_Ey| -1      | 0.0043  | -0.0061 | -0.00085 | 1.7e-05| -8.1e-07| 1.6e-06
```

5 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/resample
help model/fisher
help model/diffloglik
```