Run Bayesian Parameter Estimation

estimate_params.m

by Jaromir Benes

May 27, 2015

Summary

Use bayesian methods to estimate some of the parameters. First, set up our priors about the individual parameters, and locate the posterior mode. Then, run a posterior simulator (adaptive random-walk Metropolis) to obtain the whole distributions of the parameters.

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`, and the example database created in `read_data`. Run `read_model` and `read_data` at least once before running this m-file.

```
load read_model.mat m;
load read_data.mat d startHist endHist;
```

3 Set Up Estimation Input Structure

The estimation input struct describes which parameters to estimate and how to estimate them. A struct needs to be created with one field for each parameter that is to be estimated. Each parameter can be then assigned a cell array with up to four pieces of information:

```
E.parameter_name = {starting}
E.parameter_name = {starting,lower}
E.parameter_name = {starting,lower,upper}
E.parameter_name = {starting,lower,upper,logdist}
```

where `starting` is a starting value for the iteration, `lower` and `upper` are the lower and upper bounds, respectively, and `logdist` is a function handle taking one input and returning the log prior density.

If the starting value is `NaN`, then the currently assigned parameter value (from the model object) is used. The constants `-Inf` and `Inf` can be used for the lower and upper bounds, respectively. Use the `logdist` package to set up the log-prior function handles.

```
E = struct();

E.chi = {NaN,  0.5,  0.95,  logdist.normal(0.85,0.025)};
E.xiw = {NaN,  30,  1000,  logdist.normal(60,50)};
```

```
E.xip = {NaN,  30,  1000,  logdist.normal(300,50)};
E.rhor = {NaN,  0.10,  0.95,  logdist.beta(0.85,0.05)};
E.kappap = {NaN,  1.5,  10,  logdist.normal(3.5,1)};
E.kappan = {NaN,  0,  1, logdist.normal(0,0.2)};

E.std_Ep = {0.01,  0.001,  0.10,  logdist.invgamma(0.01,Inf)};
E.std_Ew = {0.01,  0.001,  0.10,  logdist.invgamma(0.01,Inf)};
E.std_Ea = {0.001,  0.0001,  0.01,  logdist.invgamma(0.001,Inf)};
E.std_Er = {0.005,  0.001,  0.10,  logdist.invgamma(0.005,Inf)};
E.corr_Er__Ep = {0,  -0.9,  0.9,  logdist.normal(0,0.5)};

disp(E);
```

```
         chi: {[NaN]  [0.5000]  [0.9500]  [[function_handle]]}
         xiw: {[NaN]  [30]  [1000]  [[function_handle]]}
         xip: {[NaN]  [30]  [1000]  [[function_handle]]}
        rhor: {[NaN]  [0.1000]  [0.9500]  [[function_handle]]}
      kappap: {[NaN]  [1.5000]  [10]  [[function_handle]]}
      kappan: {[NaN]  [0]  [1]  [[function_handle]]}
      std_Ep: {[0.0100]  [1.0000e-03]  [0.1000]  [[function_handle]]}
      std_Ew: {[0.0100]  [1.0000e-03]  [0.1000]  [[function_handle]]}
      std_Ea: {[1.0000e-03]  [1.0000e-04]  [0.0100]  [[function_handle]]}
      std_Er: {[0.0050]  [1.0000e-03]  [0.1000]  [[function_handle]]}
 corr_Er__Ep: {[0]  [-0.9000]  [0.9000]  [[function_handle]]}
```

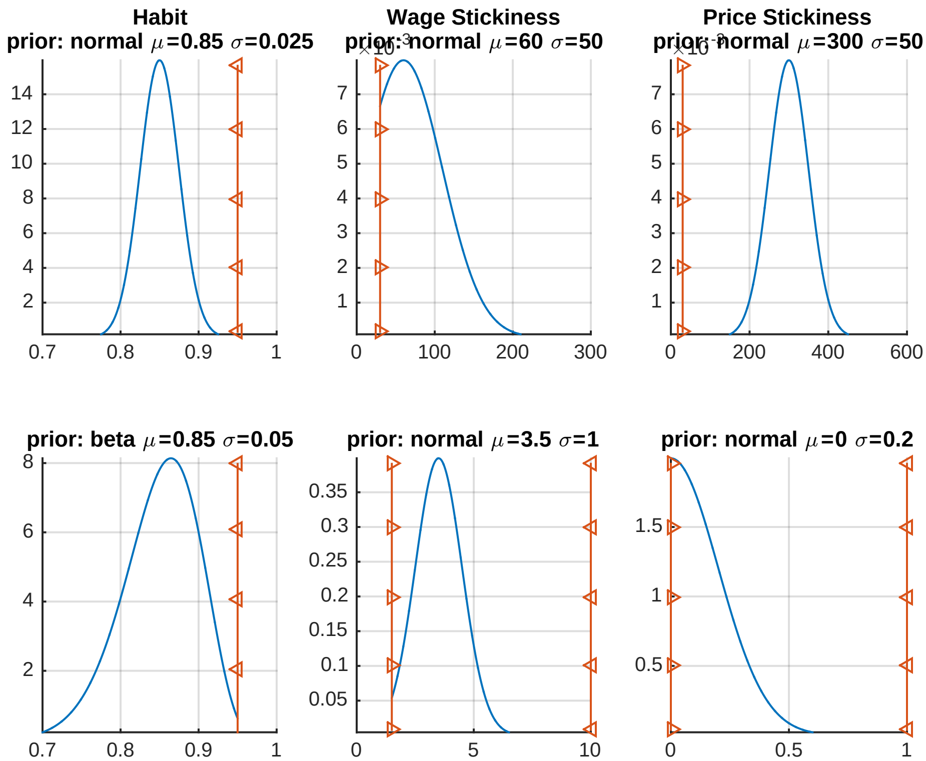
4 Visualise Prior Distributions

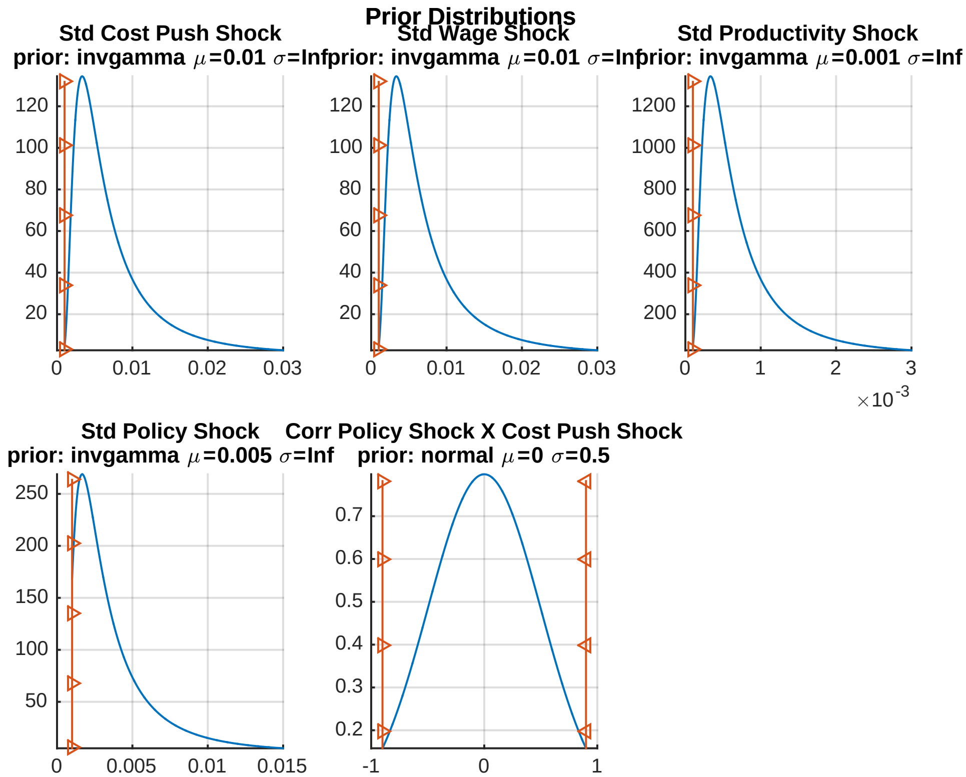
The function plotpp plots the prior distributions (this function can also plot the priors together with posteriors obtained from a posterior simulator – see below). To control the appearance and graphics properties of the various plots included in the graphs, use either the options 'figure=', 'axes=' $\boxed{1}$, 'title=' $\boxed{2}$, 'plotprior=', 'plotinit=', 'plotmode=', 'plotposter=', 'plotbounds=', or alternatively the standard Matlab function set with the graphics handles returned in the struct h.

```
c = autocaption(m,E,'$descript$');

[pr,po,h] = plotpp(E,[],[], ...
    ... 'axes=',{'fontsize=',8}, ... 1
    ... 'title=',{'fontsize=',8}, ... 2
    'subplot=',[2,3],'caption=',c); %#ok<ASGLU>

ftitle(h.figure,'Prior Distributions');
```


Habit
prior: normal μ=0.85 σ=0.025
Wage Stickiness
prior: normal μ=60 σ=50
Price Stickiness
prior: normal μ=300 σ=50
prior: beta μ=0.85 σ=0.05
prior: normal μ=3.5 σ=1
prior: normal μ=0 σ=0.2



5 Maximise Posterior Distribution to Locate Its Mode

The main output arguments are the following (these remain the same whatever the set-up of the estimation):

- `est` – Struct with point estimates.
- `pos` – Initialised posterior simulator object. The object `pos` will be used later in this file to run a posterior simulator.
- `C` – Covariance matrix of the parameter estimates based on the asymptotical hessian of the posterior density at its mode.
- `H` – Cell array 1-by-2: H1 is the hessian of the objective function returned by the Optim Tbx (should be close to `C`); H2 is a diagonal matrix with the contributions of the priors to the total hessian.
- `mest` – Model object with the new estimated parameters.

- `v` – Estimate of the common variance factor (only with the option `'relative='` `true`, which is the default setting); all the std dev of all shocks are multiplied automatically by the square root of this number.
- `delta` – Estimates of the deterministic trend parameters estimated by concentrating them out of the likelihood function.

```
filterOpt = { ...
    'outoflik=',{'Short_','Infl_','Growth_','Wage_'}, ...
    'relative=',true, ...
    };

optimSet = { ...
    'maxFunEvals=',10000, ...
    'maxIter=',100, ...
    };

tic();
[est,pos,C,H,mest,v,~,~,delta,Pdelta] = ...
    estimate(m,d,startHist:endHist,E, ...
    'filter=',filterOpt,'optimSet=',optimSet);
toc();
```

```
                                   Max     Line search  Directional  First-order
 Iter F-count          f(x)   constraint   steplength   derivative   optimality Procedure
    0      12       409.27            0
    1      26      390.185            0         0.25         -138     2.58e+04
    2      39      354.693            0          0.5        -43.3     9.24e+03
    3      52      340.247     -0.00075          0.5        -46.9     4.91e+03
    4      66      336.371   -0.0005625         0.25        -20.9     1.57e+04
    5      84      335.998   -0.0005537       0.0156        -44.2     7.07e+03
    6     101      335.576   -0.0009188       0.0312        -30.9     3.65e+03
    7     115       329.15   -0.0007087         0.25        -19.2     5.64e+03
    8     129      328.624    -0.001072         0.25        -14.8     1.01e+04
    9     143      326.323   -0.0008043         0.25        -16.8     5.56e+03
   10     158      324.633   -0.0007126        0.125        -10.4     9.42e+03
   11     174      324.243    -0.000703       0.0625         -4.7     5.82e+03
   12     188      323.447   -0.0005273         0.25        -13.3     5.52e+03
   13     203      322.677   -0.0004613        0.125         -5.3     4.86e+03
   14     217      320.276    -0.000346         0.25        -11.7        4e+03
   15     233       320.08   -0.0005059       0.0625        -5.13     4.49e+03
   16     247      319.128   -0.0003794         0.25        -4.68     3.25e+03
   17     262      318.654    -0.000332        0.125          -10          980
   18     277      318.643   -0.0003855        0.125        -1.19     2.91e+03
```

```
   19    292      318.179   -0.0003373       0.125        -2.68     6.17e+03
   20    306        318.1    -0.000253        0.25       -0.792      2.5e+03
   21    321      317.927   -0.0002503       0.125        -2.79     2.59e+03
   22    338      317.878   -0.0002831      0.0312        -2.76     2.18e+03
   23    352      317.791   -0.0002123        0.25        -1.48     1.58e+03
   24    366      317.723   -0.0002294        0.25        -1.33          538
   25    378      317.699   -0.0002119           1       -0.312          273
   26    390       317.69   -0.0002191           1       -0.119           45
   27    402      317.658   -0.0002297           1      -0.0784          495
   28    414      317.561   -0.0002496           1      -0.0643     1.32e+03
   29    426      317.349   -0.0002774           1      -0.0549     2.32e+03
   30    438      316.927   -0.0003074           1      -0.0491     3.31e+03
   31    450      316.282   -0.0003143           1      -0.0434     3.87e+03
   32    462       315.62   -0.0002653           1      -0.0371     3.24e+03
   33    474      315.353   -0.0001911           1      -0.0296        1e+03
   34    486      315.315    -0.000177           1        -1.02          306
   35    498      315.308    -0.000178           1     -0.00774         49.6
   36    510      315.308   -0.0001775           1     -0.00163         9.08
   37    522      315.307   -0.0001776           1    -0.000959         17.6  Hessian modified
   38    534      315.307   -0.0001775           1    -0.000741           11  Hessian modified
   39    546      315.307   -0.0001775           1     -0.00104         9.08  Hessian modified
   40    558      315.306   -0.0001772           1     -0.00224         35.9  Hessian modified
   41    570      315.304   -0.0001767           1     -0.00316          101  Hessian modified
   42    582        315.3   -0.0001758           1     -0.00433          183
   43    594      315.288   -0.0001744           1     -0.00509          309
   44    606      315.266   -0.0001728           1     -0.00528          401
   45    618      315.238   -0.0001723           1      -0.0052          345
   46    630      315.223   -0.0001736           1     -0.00488          157
   47    642      315.219   -0.0001747           1     -0.00397         31.2
   48    654      315.219   -0.0001748           1    -0.000724         2.07
   49    666      315.219   -0.0001748           1    -5.99e-05        0.222  Hessian modified

Local minimum possible. Constraints satisfied.

fmincon stopped because the predicted change in the objective function
is less than the selected value of the function tolerance and constraints
are satisfied to within the default value of the constraint tolerance.




No active inequalities.
Elapsed time is 15.425097 seconds.
```

6 Print Some Estimation Results

```
disp('Point estimates');
est

disp('Common variance factor');
v

disp('Out-of-lik parameters');
delta

disp('Parameters in the estimated model object');
disp('Std deviations adjusted for the common variance factor');
get(mest,'parameters')
```

```
Point estimates
est =
            chi: 0.9138
            xiw: 133.8447
            xip: 264.6905
           rhor: 0.8587
         kappap: 2.9459
         kappan: 0.3419
         std_Ep: 0.0041
         std_Ew: 0.0024
         std_Ea: 0.0014
         std_Er: 0.0012
    corr_Er__Ep: -0.1108
Common variance factor
v =
    0.6255
Out-of-lik parameters
delta =
     Short_: -3.9012
      Infl_: -0.3539
    Growth_: 0.0078
      Wage_: -1.9244
Parameters in the estimated model object
Std deviations adjusted for the common variance factor
ans =
          alpha: 1.0074
           beta: 0.9962
          gamma: 0.6000
          delta: 0.0300
              k: 10
             pi: 1.0062
```

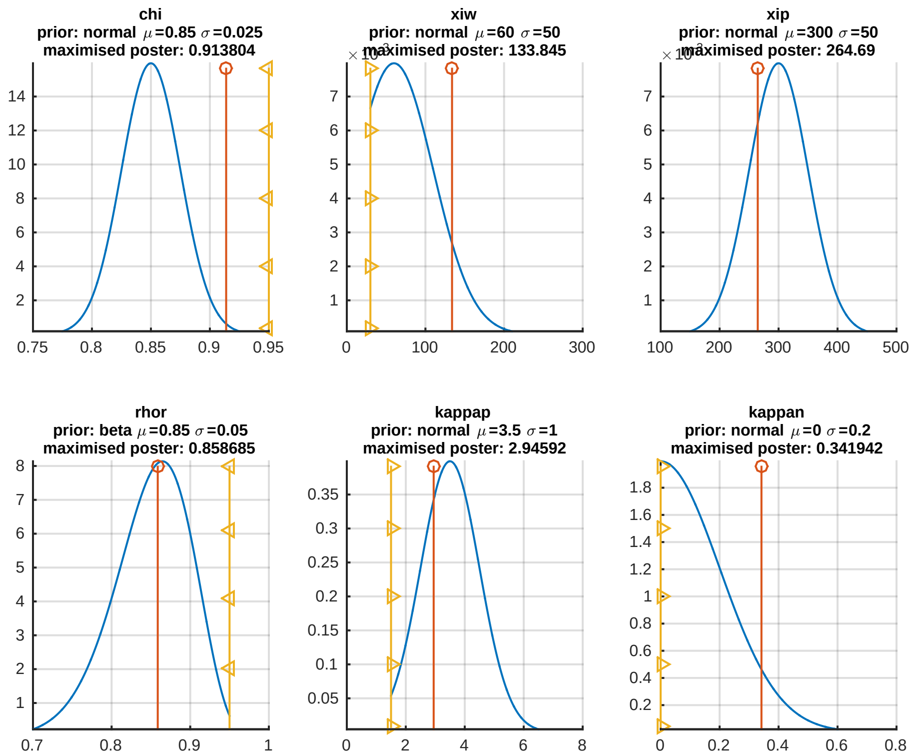
```
           eta: 6
           psi: 0.2500
           chi: 0.9138
           xiw: 133.8447
           xip: 264.6905
          rhoa: 0.9000
          rhor: 0.8587
        kappap: 2.9459
        kappan: 0.3419
        Short_: -3.9012
         Infl_: -0.3539
       Growth_: 0.0078
         Wage_: -1.9244
        std_Mp: 0
        std_Mw: 0
        std_Ey: 0.0079
        std_Ep: 0.0032
        std_Ea: 0.0011
        std_Er: 9.2918e-04
        std_Ew: 0.0019
   corr_Ep__Er: -0.1108
```

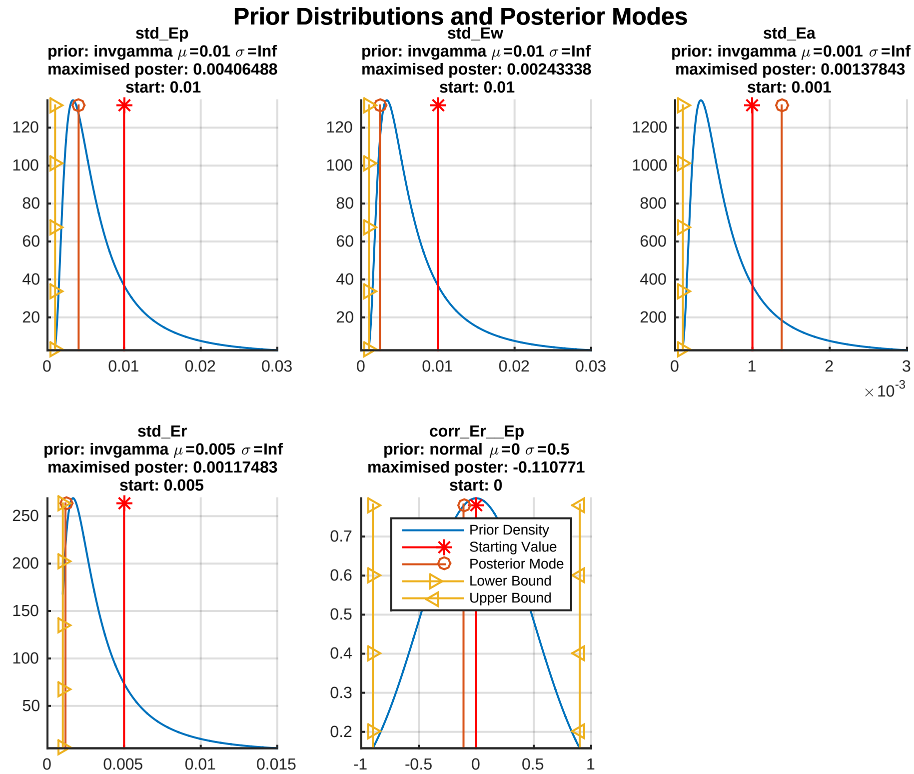
7 Visualise Prior Distributions and Posterior Modes

Use the function plotpp again supplying now the struct est with the estimated posterior modes as the second input argument. The posterior modes are added as stem graphs, and the estimated values are included in the graph titles.

```
[pr,po,h] = plotpp(E,est,[], ...
    'title=',{'fontsize=',8}, ...
    'axes=',{'fontsize=',8}, ...
    'plotInit=',{'color=','red','marker=','*'}, ...
    'subplot=',[2,3]); %#ok<ASGLU>

ftitle(h,'Prior Distributions and Posterior Modes');
legend('Prior Density','Starting Value','Posterior Mode', ...
    'Lower Bound','Upper Bound');
```


chi
prior: normal μ=0.85 σ=0.025
maximised poster: 0.913804
xiw
prior: normal μ=60 σ=50
maximised poster: 133.845
xip
prior: normal μ=300 σ=50
maximised poster: 264.69
rhor
prior: beta μ=0.85 σ=0.05
maximised poster: 0.858685
kappap
prior: normal μ=3.5 σ=1
maximised poster: 2.94592
kappan
prior: normal μ=0 σ=0.2
maximised poster: 0.341942



8 User Supplied Optimisation Routine

Uncomment the block of code below in order to get it executed.

Set up the `estimate` command with a user-supplied optimisation routine. Re-use the Optim Tbx's functions to illustrate the implementation details. Fell free though to use any kind of third-party solver.

Proceed in two steps:

1. Write a `mymin` m-file to organise the input and output arguments as required by the `estimate` function. Note also that an extra input argument with the settings will be passed in.
2. Call the `estimate` function and pass in a function handle to `mymin` through the option `'solver'`. Because the same routine is effective used as before, the results ought to be identical (although the execution might be somewhat slower).

```
%{
% edit mymin.m;

tic();
[est1,pos1,C1,H1,mest1,v1,ans,ans,delta1,Pdelta1] = ...
    estimate(m,d,startHist:endHist,E, ...
    'filter=',filterOpt, ...
    'solver=',@mymin); %#ok<NOANS,ASGLU>
toc();

dbfun(@(x,y) [x,y,y-x],est,est1)
%}
```

9 Covariance Matrix of Parameter Estimates

Compute the std deviations of the parameter estimates by taking the square roots of the diagonal entries in the Hessian returned by the optimisation routine.

```
plist = fieldnames(E);

std = sqrt(diag(inv(H{1})));

disp('Std deviations of parameter estimates');
[char(plist), num2str(std,': %-g') ]
```

```
Std deviations of parameter estimates
ans =
chi        : 0.0189469
xiw        : 39.9031
xip        : 48.452
rhor       : 0.0295335
kappap     : 1.1377
kappan     : 0.089874
std_Ep     : 0.00056756
std_Ew     : 0.000405932
std_Ea     : 0.000306453
std_Er     : 0.000183524
corr_Er__Ep: 0.148394
```

10 Examine Neighbourhood Around Optimum

The function neighbourhood evaluates the posterior density (accessible through the poster object pos) at a number of points around the optimum for each parameter. In the code below, each parameter estimate is examined within the range of +/- 5 % of the posterior mode (i.e., 0.95 : 0.01 : 1.05 times the value of the estimate).

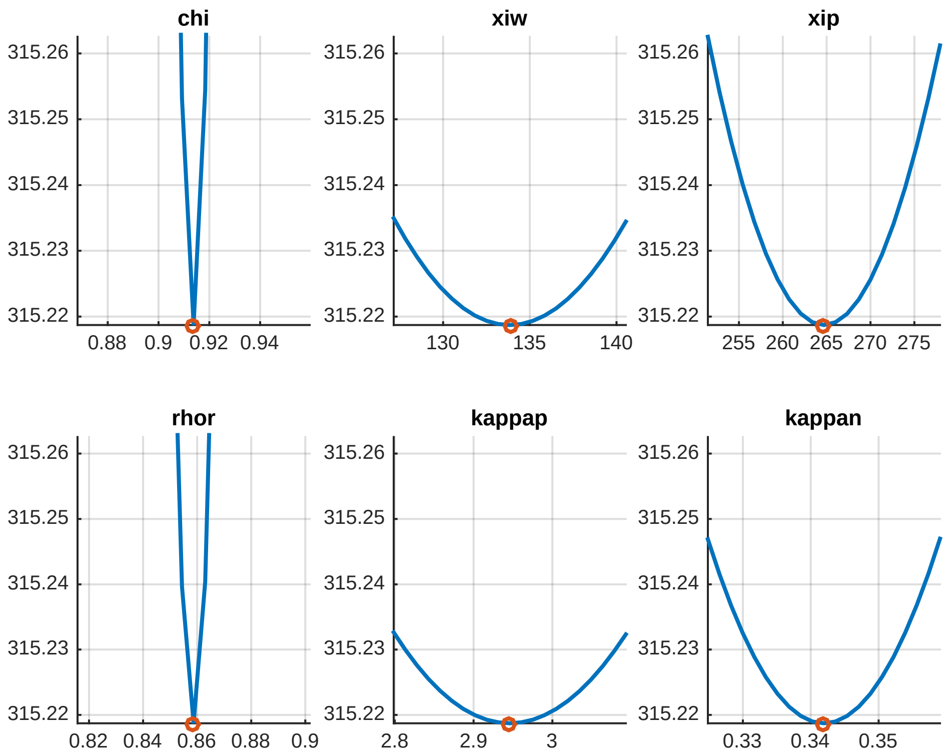
The plotneigh function then plots graphs depicting the local behaviour of both the overall objective funtion (minus log posterior density) and the data likelihood (minus log likelihood). Note that the likelihood curve is shifted up or down by an arbitrary constant to make it fit in the graph.

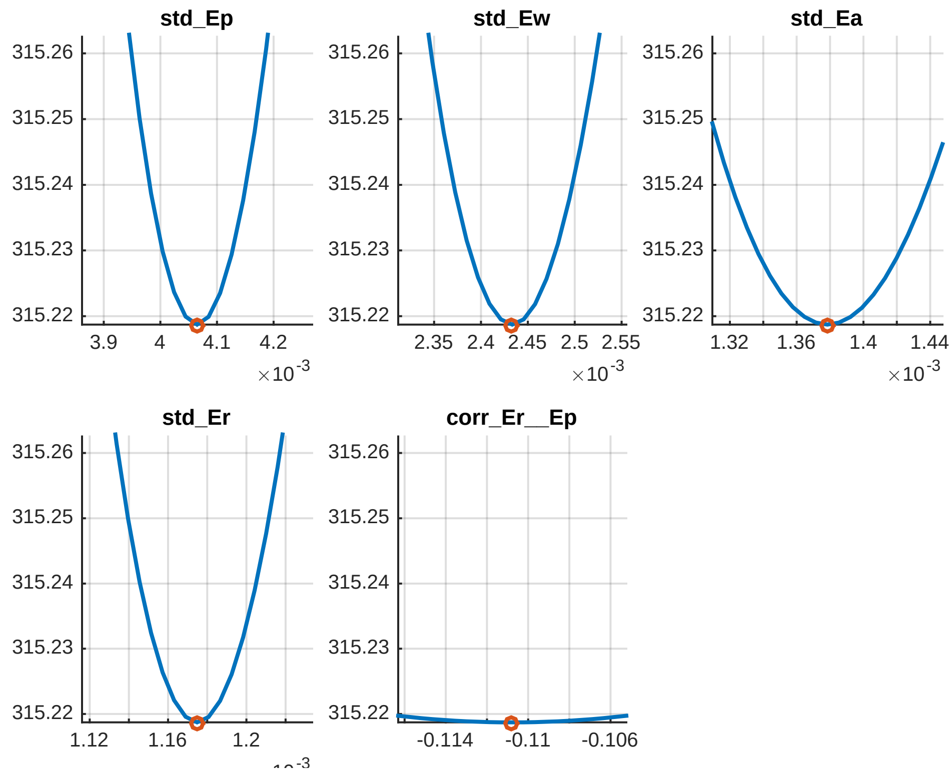
The option 'linkaxes' makes the y-axes identical in all graphs to help compare the curvature of the posterior density around the individual parameter estimates. This indicates the degree of identification.

```
n = neighbourhood(mest,pos,0.95:0.005:1.05, ...
    'progress=',true,'plot=',false)

plotneigh(n,'linkaxes=',true,'subplot=',[2,3], ...
    'plotobj=',{'linewidth=',2}, ...
    'plotest=',{'marker=','o','linewidth=',2}, ...
    'plotbounds=',{'lineStyle','--','lineWidth',2});
```

```
[--IRIS model.neighbourhood progress-----]
[****************************************]
n =
            chi: {1x3 cell}
            xiw: {[21x1 double]  [21x2 double]  [133.8447 315.2187 30 1000]}
            xip: {[21x1 double]  [21x2 double]  [264.6905 315.2187 30 1000]}
           rhor: {1x3 cell}
         kappap: {[21x1 double]  [21x2 double]  [2.9459 315.2187 1.5000 10]}
         kappan: {[21x1 double]  [21x2 double]  [0.3419 315.2187 0 1]}
         std_Ep: {[21x1 double]  [21x2 double]  [1x4 double]}
         std_Ew: {[21x1 double]  [21x2 double]  [1x4 double]}
         std_Ea: {[21x1 double]  [21x2 double]  [1x4 double]}
         std_Er: {[21x1 double]  [21x2 double]  [1x4 double]}
    corr_Er__Ep: {[21x1 double]  [21x2 double]  [1x4 double]}
```


chi
xiw
xip
rhor
kappap
kappan
315.26
315.25
315.24
315.23
315.22
0.88 0.9 0.92 0.94
130 135 140
255 260 265 270 275
0.82 0.84 0.86 0.88 0.9
2.8 2.9 3
0.33 0.34 0.35



11 Run Metropolis Random Walk Posterior Simulator

Run 5,000 draws from the posterior distribution using an adaptive version of the random-walk Metropolis algorithm. The number of draws, `N=1000`, should be obviously much larger in practice (such as 100,000 or 1,000,000). Use then the function `stats` to calculate some statistics of the simulated parameter chains – by default, the simulated chains, their means, std errors, high probability density intervals, and the marginal data density are returned. Feel free to change the list of requested characteristics; see help on `poster/stats` for details.

The output argument `ar` monitors the evolution of the acceptance ratio. The default target acceptance ratio is 0.234 (can be modified using the option `'targetAR'` in `arwm`), the covariance of the proposal distribution is gradually adapted to achieve this target.

```
N = 1000

tic;
[theta,logpost,ar] = arwm(pos,N, ...
    'progress=',true,'adaptScale=',2,'adaptProposalCov=',1,'burnin=',0.20);
```

```
toc;

disp('Final acceptance ratio');
ar(end)

s = stats(pos,theta,logpost)
```

```
N =
        1000
[--IRIS poster.arwm progress-------------]
[*****************************************]
Elapsed time is 22.843962 seconds.
Final acceptance ratio
ans =
    0.2390
s =
    chain: [1x1 struct]
     mean: [1x1 struct]
      std: [1x1 struct]
     hist: [1x1 struct]
      mdd: -343.3406
```

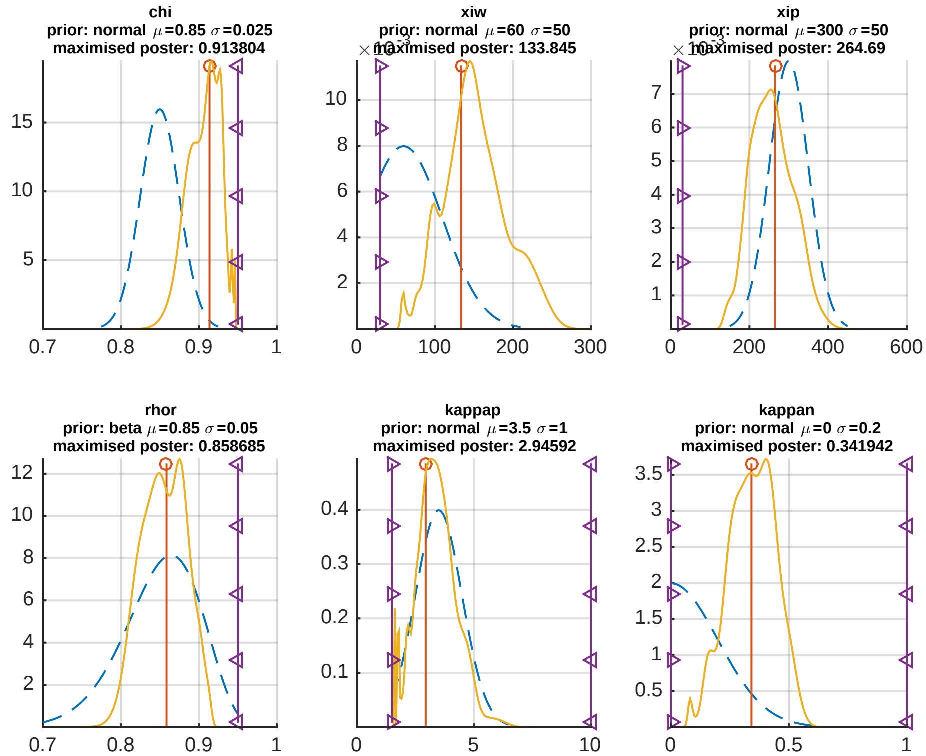
12 Visualise Priors and Posteriors

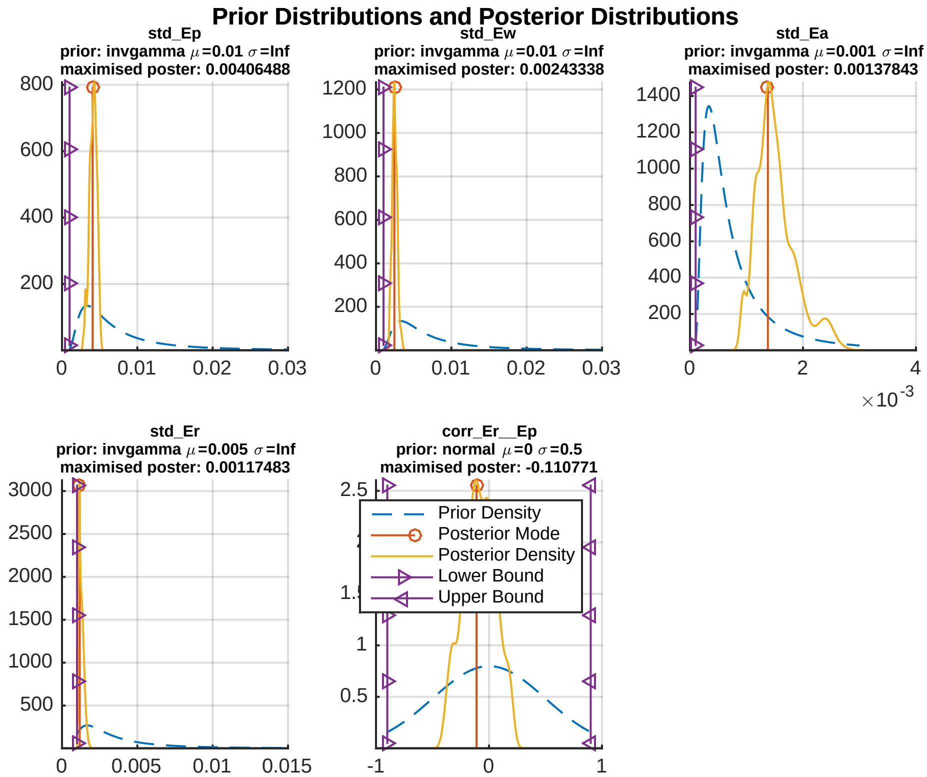
Because the number of draws from the posterior distribution is very low, `N=1000`, the posterior graphs are far from being smooth, and may visibly change if another posterior chain is generated.

```
[pr,po,h] = plotpp(E,est,theta, ...
    'plotprior=',{'linestyle=','--'}, ...
    'title=',{'fontsize=',8}, ...
    'subplot=',[2,3]);

ftitle(h.figure,'Prior Distributions and Posterior Distributions');

legend('Prior Density','Posterior Mode','Posterior Density', ...
    'Lower Bound','Upper Bound');
```


chi
prior: normal μ=0.85 σ=0.025
maximised poster: 0.913804
xiw
prior: normal μ=60 σ=50
maximised poster: 133.845
xip
prior: normal μ=300 σ=50
maximised poster: 264.69
rhor
prior: beta μ=0.85 σ=0.05
maximised poster: 0.858685
kappap
prior: normal μ=3.5 σ=1
maximised poster: 2.94592
kappan
prior: normal μ=0 σ=0.2
maximised poster: 0.341942



13 Save Model Object with Estimated Parameters

```
save estimate_params.mat mest pos E theta logpost;
```

14 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/estimate
help model/neighbourhood
help poster/arwm
help poster/stats
help grfun/plotpp
help logdist
help logdist.normal
```

```
help logdist.lognormal
help logdist.beta
help logdist.gamma
help logdist.invgamma
help logdist.uniform
```