Run bayesian parameter estimation

[estimate_params.m]

by

Jaromir Benes

20 May 2011

Introduction

In this m-file, we use bayesian methods to estimate some of the parameters. First, we set up our priors about the individual parameters. Second, we locate the posterior mode. Third, we run a posterior simulator (adaptive random-walk Metropolis) to obtain the whole distributions of the parameters.

Contents

1 Check IRIS version

```
irisrequired(8.20110412);
```

2 Clear workspace and load model, data, and dates

```
home();
clear();
close('all');
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. We need to create a struct 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,logprior}
```

where `starting` is a starting value for the iteration, `lower` and `upper` are the lower and upper bounds, respectively, and `logprior` 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. You can use `-Inf` and `Inf` for the lower and upper bounds, respectively. You can use the `logprior` package to set up the log-prior function handles.

```
E = struct();

E.chi = {NaN, 0.5, 0.95, logprior.normal(0.85,0.025)};
E.xiw = {NaN, 10, 1000, logprior.normal(60,50)};
E.xip = {NaN, 10, 1000, logprior.normal(300,50)};
E.rhor = {NaN, 0.10, 0.95, logprior.beta(0.85,0.05)};
E.kappap = {NaN, 1.5, 10, logprior.normal(3.5,1)};
E.kappan = {NaN, 0, 1, logprior.normal(0,0.2)};

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

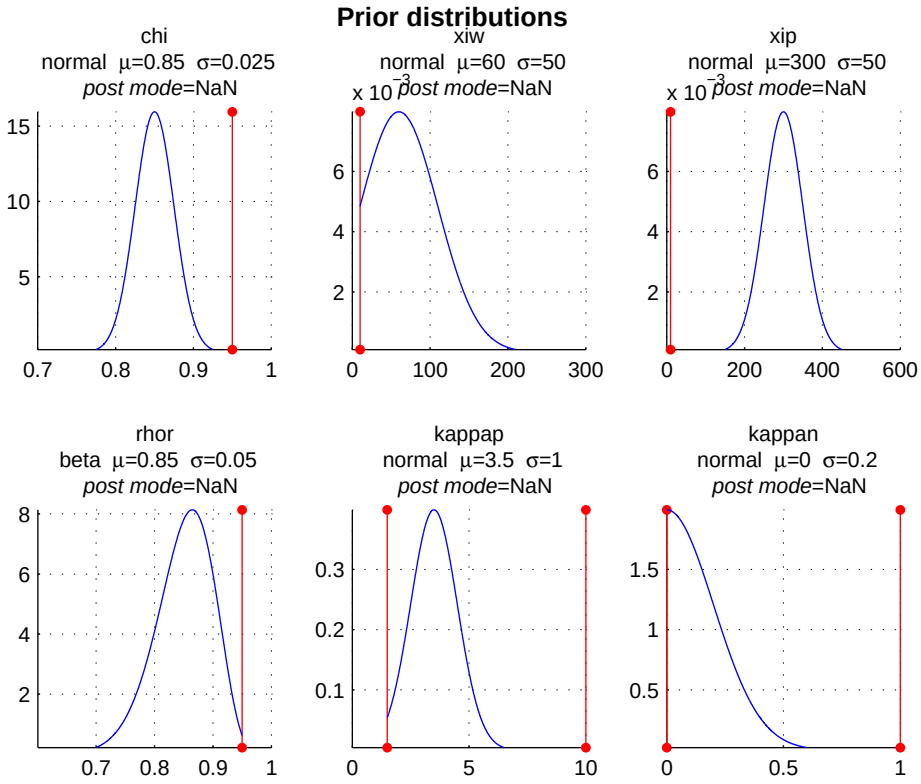
```
57
58 E
```

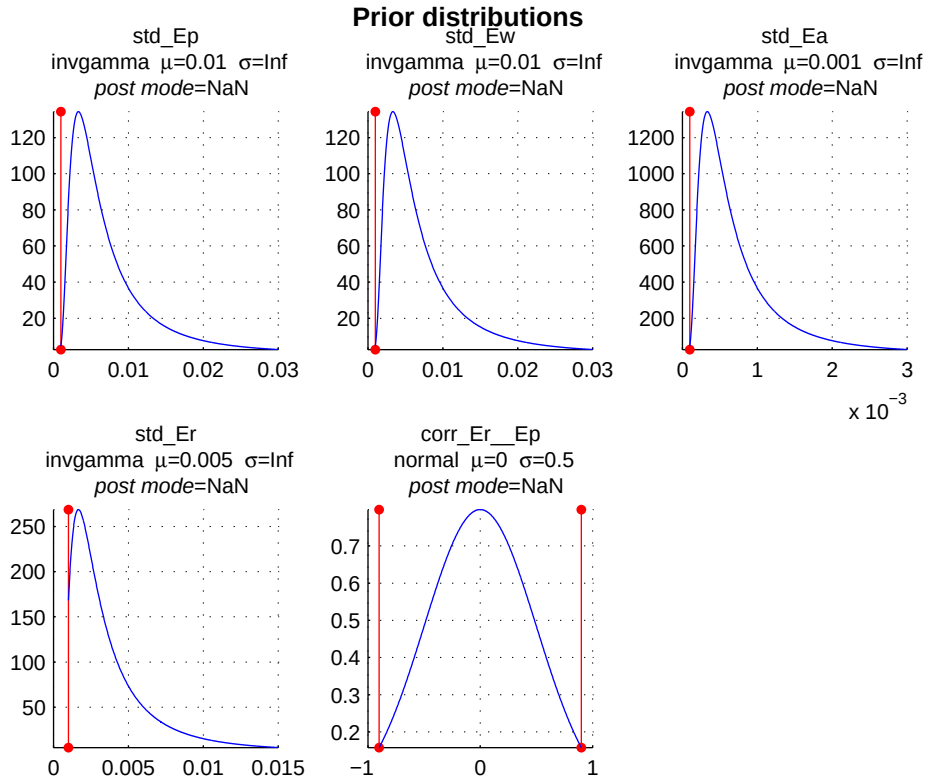
```
E =
           chi: {[NaN] [0.5000] [0.9500] [[function_handle]]}
           xiw: {1x4 cell}
           xip: {1x4 cell}
          rhor: {[NaN] [0.1000] [0.9500] [[function_handle]]}
        kappap: {[NaN] [1.5000] [10] [[function_handle]]}
        kappan: {[NaN] [0] [1] [@(x,varargin)normal_(x,a,b,mu,sgm,mode,varargin{:})]}
        std_Ep: {[0.0100] [1.0000e-003] [0.1000] [[function_handle]]}
        std_Ew: {[0.0100] [1.0000e-003] [0.1000] [[function_handle]]}
        std_Ea: {[1.0000e-003] [1.0000e-004] [0.0100] [[function_handle]]}
        std_Er: {[0.0050] [1.0000e-003] [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). The output arguments give you access to the graphics objects created – figures, axes, line plots, titles.

```
67 [pr,po,fig,ax,prlin,polin,blin,tit] = plotpp(E,[],'subplot',[2,3]); %#ok<ASGLU>
68
69 ftitle(fig,'Prior distributions');
70
71 set(blin,'marker','.','markerSize',11);
72 set([ax,tit],'fontSize',8);
```


Prior distributions
chi
normal μ=0.85 σ=0.025
post mode=NaN
xiw
normal μ=60 σ=50
post mode=NaN
x 10^-3
xip
normal μ=300 σ=50
post mode=NaN
x 10^-3
rhor
beta μ=0.85 σ=0.05
post mode=NaN
kappap
normal μ=3.5 σ=1
post mode=NaN
kappan
normal μ=0 σ=0.2
post mode=NaN



5 Find posterior 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. You can use pos
- `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 (if you run it with 'relative' set to true = default); 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.

```
[est,pos,C,H,mest,v,ans,ans,delta,Pdelta] = estimate(m,d,starthist:endhist,E, ...
    'outoflik',{'Short_','Infl_','Growth_','Wage_'},'relative',false); ...
    %#ok<ASGLU,NOANS>

est
v
delta
get(mest,'parameters')
```

```
                          Max    Line search Directional First-order
 Iter F-count      f(x)  constraint  steplength  derivative  optimality Procedure
    0      12     485.031           0
    1      25       437.9           0         0.5       -76.2   1.47e+004
    2      41     435.415           0      0.0625       -32.9   1.32e+004
    3      55     408.353           0        0.25       -73.8   1.07e+004
    4      68     364.805           0         0.5       -56.1   3.48e+004
    5      81     348.647           0         0.5        -319   1.96e+004
    6     100      347.35  -0.0005429     0.00781       -96.8   5.76e+004
    7     115     343.095  -0.0004751       0.125       -30.2   3.39e+004
    8     131     342.294  -0.0006483      0.0625       -62.7   1.65e+004
    9     144     328.047  -0.0003242         0.5       -21.4   1.03e+004
   10     158     325.469  -0.0002431        0.25       -8.26   8.87e+003
   11     174     324.352  -0.0002279      0.0625       -23.8   5.36e+003
   12     190      323.33  -0.0002137      0.0625       -17.9   2.14e+003
   13     204     323.115  -0.0001603        0.25       -15.1   4.38e+003
   14     218      322.88  -0.0001202        0.25       -3.15    4.2e+003
   15     232     322.747 -9.015e-005        0.25       -12.8   7.75e+003
   16     246      322.39 -6.761e-005        0.25        -2.7   8.93e+003
   17     263     322.075  -0.0001324      0.0313       -3.55   2.52e+003
   18     279         322  -0.0001242      0.0625       -1.38   4.63e+003
   19     294     321.616  -0.0001086       0.125        -6.5   3.76e+003
   20     309     321.575 -9.507e-005       0.125       -1.95   1.57e+003
   21     322     321.339 -4.753e-005         0.5       -3.27   3.35e+003
   22     338     321.325 -7.052e-005      0.0625      -0.907        603
   23     354     321.311   -7.1e-005      0.0625      -0.434   1.23e+003
   24     373     321.309  -7.14e-005     0.00781     -0.0947   1.09e+003
   25     385     321.286 -5.853e-005           1      -0.889        116
   26     397     321.281  -5.92e-005           1     -0.0782         99
   27     409     321.253 -6.184e-005           1     -0.0611        207
   28     421     321.206 -6.408e-005           1     -0.0453        406
   29     433     321.072 -6.713e-005           1     -0.0425        725
```

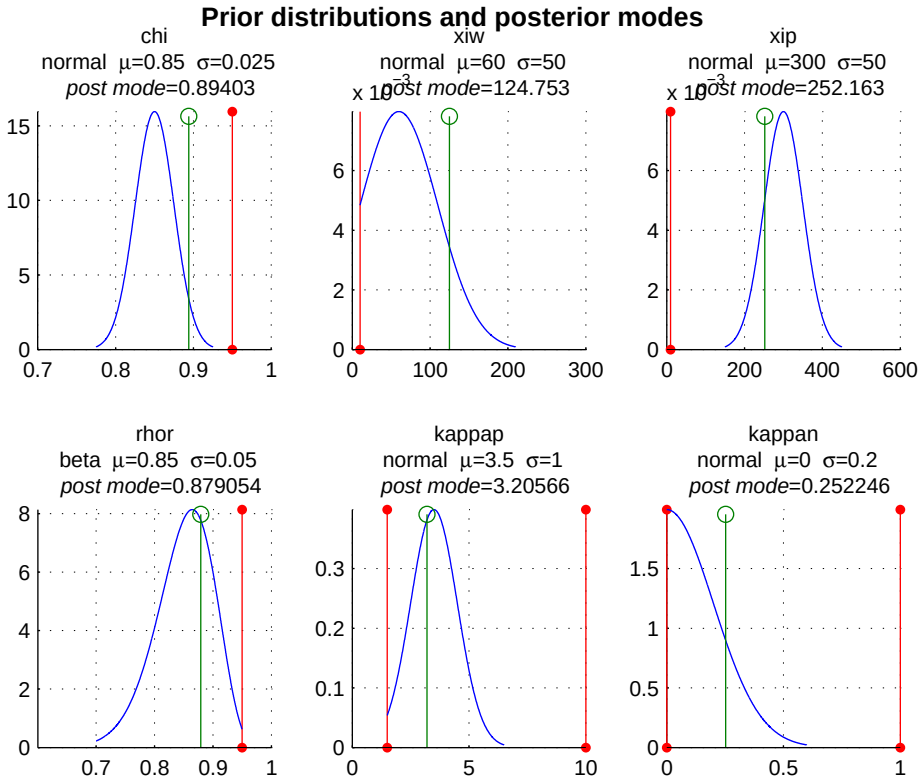
```
   30    445     320.798 -6.807e-005          1      -0.039   1.03e+003
   31    457     320.287 -6.084e-005          1     -0.0365   1.03e+003
   32    469     319.737 -3.819e-005          1     -0.0313         645
   33    481     319.537 -1.958e-005          1      -0.022         377
   34    493     319.507 -1.857e-005          1     -0.0301         193
   35    505       319.5 -1.938e-005          1    -0.00643        43.3
   36    517     319.499 -2.005e-005          1    -0.00128        19.1
   37    529     319.499  -1.99e-005          1   -0.000613        15.4
   38    541     319.499 -1.972e-005          1   -0.000996        3.68  Hessian modified
   39    553     319.498 -1.938e-005          1    -0.00165        34.1  Hessian modified
   40    565     319.497 -1.877e-005          1    -0.00285         100  Hessian modified
   41    577     319.493 -1.783e-005          1    -0.00414         214  Hessian modified
   42    589     319.483 -1.637e-005          1    -0.00534         395
   43    601     319.463 -1.461e-005          1    -0.00591         614
   44    613     319.427  -1.35e-005          1    -0.00599         774
   45    625     319.387  -1.48e-005          1    -0.00571         643
   46    637     319.368 -1.776e-005          1    -0.00547         265
   47    649     319.365 -1.942e-005          1    -0.00416        37.9
   48    661     319.365 -1.971e-005          1   -0.000468        2.33
   49    673     319.365 -1.972e-005          1  -5.63e-005       0.732  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.
est =
           chi: 0.8940
           xiw: 124.7535
           xip: 252.1627
          rhor: 0.8791
        kappap: 3.2057
        kappan: 0.2522
        std_Ep: 0.0032
        std_Ew: 0.0020
        std_Ea: 0.0010
        std_Er: 0.0010
    corr_Er__Ep: -0.1722
v =
     1
delta =
     Short_: -3.9827
```

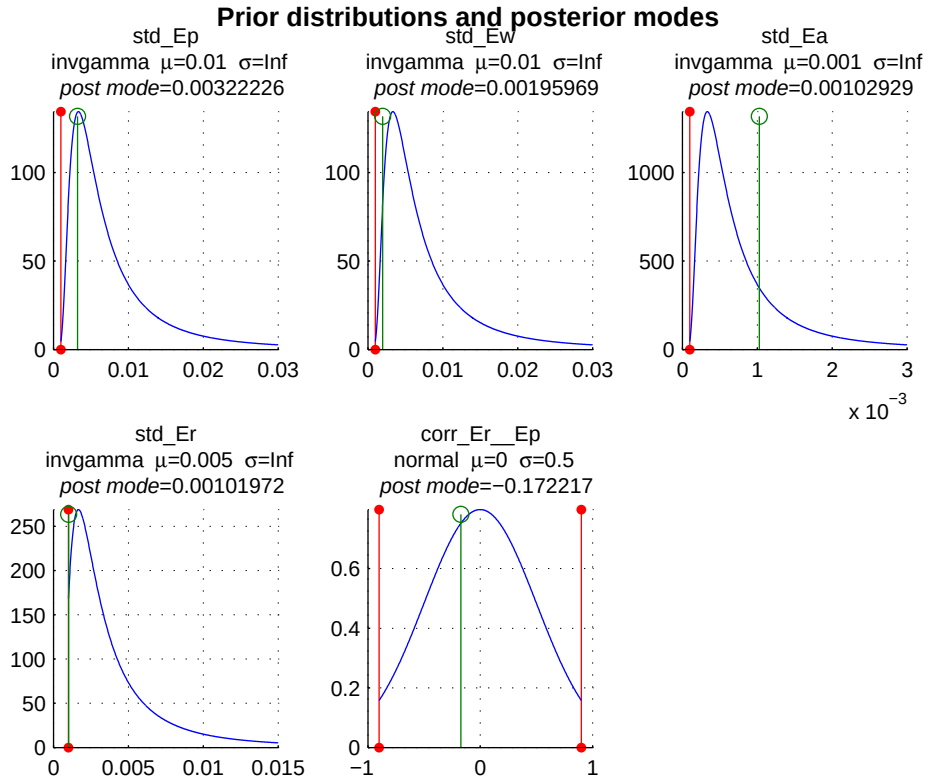
```
      Infl_: -0.3358
    Growth_: 0.0817
      Wage_: -2.0143
ans =
         alpha: 1.0074
          beta: 0.9962
         gamma: 0.6000
         delta: 0.0300
             k: 10
            pi: 1.0062
           eta: 6
           psi: 0.2500
           chi: 0.8940
           xiw: 124.7535
           xip: 252.1627
          rhoa: 0.9000
          rhor: 0.8791
        kappap: 3.2057
        kappan: 0.2522
        Short_: -3.9827
         Infl_: -0.3358
       Growth_: 0.0817
         Wage_: -2.0143
        std_Mp: 0
        std_Mw: 0
        std_Ey: 0.0100
        std_Ep: 0.0032
        std_Ea: 0.0010
        std_Er: 0.0010
        std_Ew: 0.0020
    corr_Ep__Er: -0.1722
```

6 Visualise prior distributions and posterior modes

We again use the `plotpp` function 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.

```
109 [pr,po,fig,ax,prlin,polin,blin,tit] = plotpp(E,est,'subplot',[2,3]);
110
111 ftitle(fig,'Prior distributions and posterior modes');
112
113 set(blin,'marker','.','markerSize',11);
114 set([ax,tit],'fontSize',8);
```


Prior distributions and posterior modes
chi
normal μ=0.85 σ=0.025
post mode=0.89403
xiw
normal μ=60 σ=50
post mode=124.753
x 10^-3
xip
normal μ=300 σ=50
post mode=252.163
x 10^-3
rhor
beta μ=0.85 σ=0.05
post mode=0.879054
kappap
normal μ=3.5 σ=1
post mode=3.20566
kappan
normal μ=0 σ=0.2
post mode=0.252246



7 User-supplied optimisation routine

Here we show how to set up the `estimate` command with a user-supplied optimisation routine. We re-use the Optim Tbx's functions to illustrate the implementation details. You can obviously use any kind of third-party solver.

First, we write a `myfunc` m-file to organise the input and output arguments as required by the `estimate` function. Note also that we will be passing in an extra input argument with the settings.

Second, we call the `estimate` function and pass in a function handle to `my_func` through the option `'solver'`. Because we are effectively using the same routine as before, the results ought to be identical.

```
131 edit('myfunc.m');
132
133 [est1,pos1,C1,H1,mest1,v1,ans,ans,delta1,Pdelta1] = estimate(m,d,starthist:endhist,E, ...
134     'outoflik',{'Short_','Infl_','Growth_','Wage_'},'relative',false, ...
135     'solver',@myfunc); %#ok<NOANS,ASGLU>
136
137 est & est1
```

Iter	F-count	f(x)	Max constraint	Line search steplength	Directional derivative	First-order optimality	Procedure
0	12	485.031	0				
1	25	437.9	0	0.5	-76.2	1.47e+004	
2	41	435.415	0	0.0625	-32.9	1.32e+004	
3	55	408.353	0	0.25	-73.8	1.07e+004	
4	68	364.805	0	0.5	-56.1	3.48e+004	
5	81	348.647	0	0.5	-319	1.96e+004	
6	100	347.35	-0.0005429	0.00781	-96.8	5.76e+004	
7	115	343.095	-0.0004751	0.125	-30.2	3.39e+004	
8	131	342.294	-0.0006483	0.0625	-62.7	1.65e+004	
9	144	328.047	-0.0003242	0.5	-21.4	1.03e+004	
10	158	325.469	-0.0002431	0.25	-8.26	8.87e+003	
11	174	324.352	-0.0002279	0.0625	-23.8	5.36e+003	
12	190	323.33	-0.0002137	0.0625	-17.9	2.14e+003	
13	204	323.115	-0.0001603	0.25	-15.1	4.38e+003	
14	218	322.88	-0.0001202	0.25	-3.15	4.2e+003	
15	232	322.747	-9.015e-005	0.25	-12.8	7.75e+003	
16	246	322.39	-6.761e-005	0.25	-2.7	8.93e+003	
17	263	322.075	-0.0001324	0.0313	-3.55	2.52e+003	
18	279	322	-0.0001242	0.0625	-1.38	4.63e+003	
19	294	321.616	-0.0001086	0.125	-6.5	3.76e+003	
20	309	321.575	-9.507e-005	0.125	-1.95	1.57e+003	
21	322	321.339	-4.753e-005	0.5	-3.27	3.35e+003	
22	338	321.325	-7.052e-005	0.0625	-0.907	603	
23	354	321.311	-7.1e-005	0.0625	-0.434	1.23e+003	
24	373	321.309	-7.14e-005	0.00781	-0.0947	1.09e+003	
25	385	321.286	-5.853e-005	1	-0.889	116	
26	397	321.281	-5.92e-005	1	-0.0782	99	
27	409	321.253	-6.184e-005	1	-0.0611	207	
28	421	321.206	-6.408e-005	1	-0.0453	406	
29	433	321.072	-6.713e-005	1	-0.0425	725	
30	445	320.798	-6.807e-005	1	-0.039	1.03e+003	
31	457	320.287	-6.084e-005	1	-0.0365	1.03e+003	
32	469	319.737	-3.819e-005	1	-0.0313	645	
33	481	319.537	-1.958e-005	1	-0.022	377	
34	493	319.507	-1.857e-005	1	-0.0301	193	
35	505	319.5	-1.938e-005	1	-0.00643	43.3	
36	517	319.499	-2.005e-005	1	-0.00128	19.1	
37	529	319.499	-1.99e-005	1	-0.000613	15.4	
38	541	319.499	-1.972e-005	1	-0.000996	3.68	Hessian modified
39	553	319.498	-1.938e-005	1	-0.00165	34.1	Hessian modified
40	565	319.497	-1.877e-005	1	-0.00285	100	Hessian modified
41	577	319.493	-1.783e-005	1	-0.00414	214	Hessian modified
42	589	319.483	-1.637e-005	1	-0.00534	395	

```
   43    601     319.463 -1.461e-005          1    -0.00591        614
   44    613     319.427  -1.35e-005          1    -0.00599        774
   45    625     319.387  -1.48e-005          1    -0.00571        643
   46    637     319.368 -1.776e-005          1    -0.00547        265
   47    649     319.365 -1.942e-005          1    -0.00416       37.9
   48    661     319.365 -1.971e-005          1   -0.000468       2.33
   49    673     319.365 -1.972e-005          1   -5.63e-005      0.732 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.
ans =
           chi: [0.8940 0.8940]
           xiw: [124.7535 124.7535]
           xip: [252.1627 252.1627]
          rhor: [0.8791 0.8791]
        kappap: [3.2057 3.2057]
        kappan: [0.2522 0.2522]
        std_Ep: [0.0032 0.0032]
        std_Ew: [0.0020 0.0020]
        std_Ea: [0.0010 0.0010]
        std_Er: [0.0010 0.0010]
    corr_Er__Ep: [-0.1722 -0.1722]
```

8 Covariance matrix of the parameter estimates

We compare the covariance matrix of the parameter estimates based on the asymptotic hessian of the posterior density (this covariance is returned in `C`) and the covariance matrix based on the hessian used by the Optimization Toolbox in the last iteration (this hessian is returned in `H1`).

The matrix `C` is also used as the initial proposal covariance matrix in the posterior simulator.

```
150 plist = fieldnames(E);
151
152 std1 = sqrt(diag(C));
153 std2 = sqrt(diag(inv(H{1})));
154
155 disp('Compare implied std devs of parameter estimates');
156 disp('1st column is based on asumptotic hessian');
157 disp('2nd column is based on Optim Tbx hessian in final iteration');
```

```
158 [ char(plist), num2str(std1,'| %-g'), num2str(std2,'| %-g') ]
```

```
Compare implied std devs of parameter estimates
1st column is based on asumptotic hessian
2nd column is based on Optim Tbx hessian in final iteration
ans =
chi       | 0.0182378  | 0.0183576
xiw       | 39.7255    | 38.3791
xip       | 47.0246    | 55.4894
rhor      | 0.0271404  | 0.0257316
kappap    | 0.842938   | 0.90429
kappan    | 0.0939489  | 0.0903314
std_Ep    | 0.000289214 | 0.000280623
std_Ew    | 0.000237815 | 0.000240617
std_Ea    | 0.00027826 | 0.000208971
std_Er    | 9.19013e-005| 0.000123288
corr_Er__Ep| 0.164681  | 0.143521
```

9 Examine the neighbourhood around the optimum

The `neighbourhood` function evaluates the posterior density (accessible through the poster object `pos`) for a few values of each parameter around the optimum. In this case, each parameter estimate will investigate within the range of +/- 5 % of the posterior mode (i.e., `0.95 : 0.01 : 1.05` times the value of the estimate).

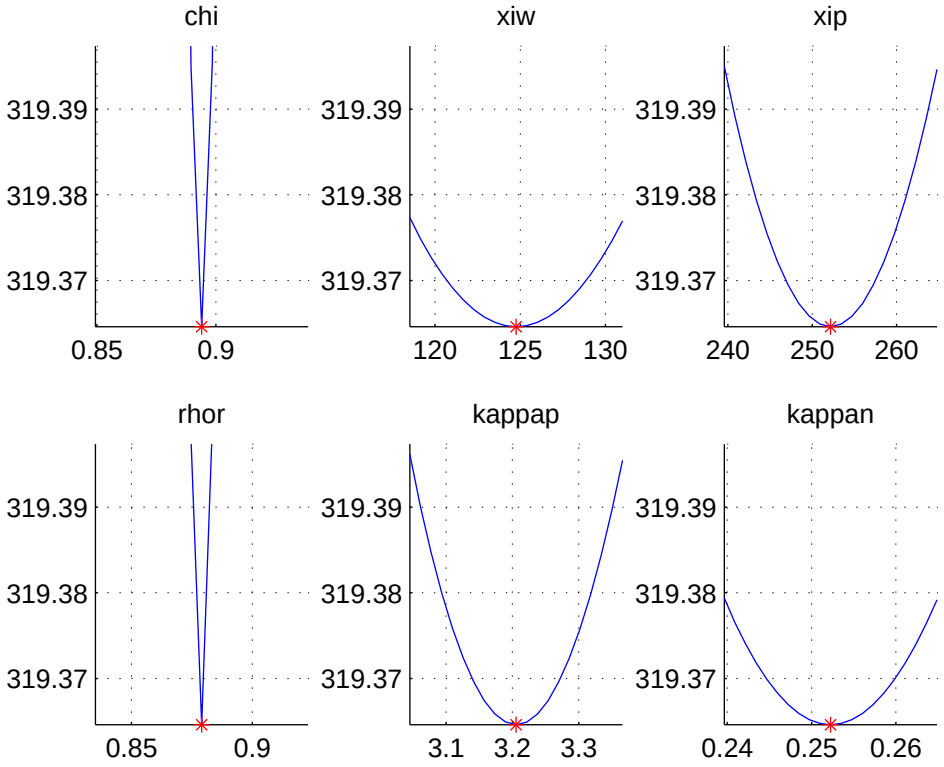
The `plotneigh` function then plots graphs depicting the local behaviour of both the overall objective funtion (minus posterior log density) and the data likelihood (minus log likelihood). Note that the likelihood curve is shifted up or down by an arbitrary constant to make it fit in the graph.

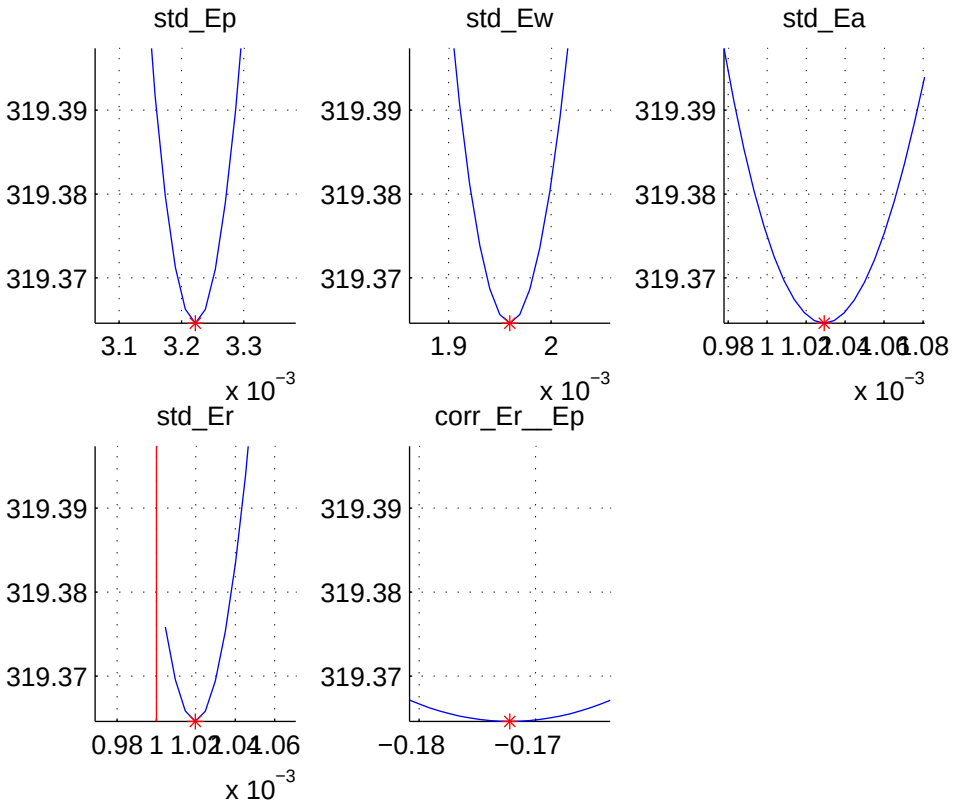
The option `'linkaxes'` makes the y-axes identical in all graphs to help compare the curvature of the posterior density around the individual parameter estimates. This indicates the degree of identification.

```
178 n = neighbourhood(mest,pos,0.95:0.005:1.05,'progress',true,'plot',false)
179
180 [fig,ax,objh,likh,esth,bh] = plotneigh(n,'linkaxes',true,'subplot',[2,3]); %#ok<ASGLU>
```

```
[--IRIS model.neighbourhood progress-----]
[****************************************]
n =
           chi: {[21x1 double] [21x2 double] [0.8940 319.3646 0.5000 0.9500]}
           xiw: {[21x1 double] [21x2 double] [124.7535 319.3646 10 1000]}
           xip: {[21x1 double] [21x2 double] [252.1627 319.3646 10 1000]}
          rhor: {[21x1 double] [21x2 double] [0.8791 319.3646 0.1000 0.9500]}
        kappap: {[21x1 double] [21x2 double] [3.2057 319.3646 1.5000 10]}
        kappan: {[21x1 double] [21x2 double] [0.2522 319.3646 0 1]}
```

```
        std_Ep: {[21x1 double] [21x2 double] [0.0032 319.3646 1.0000e-003 0.1000]}
        std_Ew: {[21x1 double] [21x2 double] [0.0020 319.3646 1.0000e-003 0.1000]}
        std_Ea: {[21x1 double] [21x2 double] [0.0010 319.3646 1.0000e-004 0.0100]}
        std_Er: {[21x1 double] [21x2 double] [0.0010 319.3646 1.0000e-003 0.1000]}
   corr_Er__Ep: {[21x1 double] [21x2 double] [-0.1722 319.3646 -0.9000 0.9000]}
```

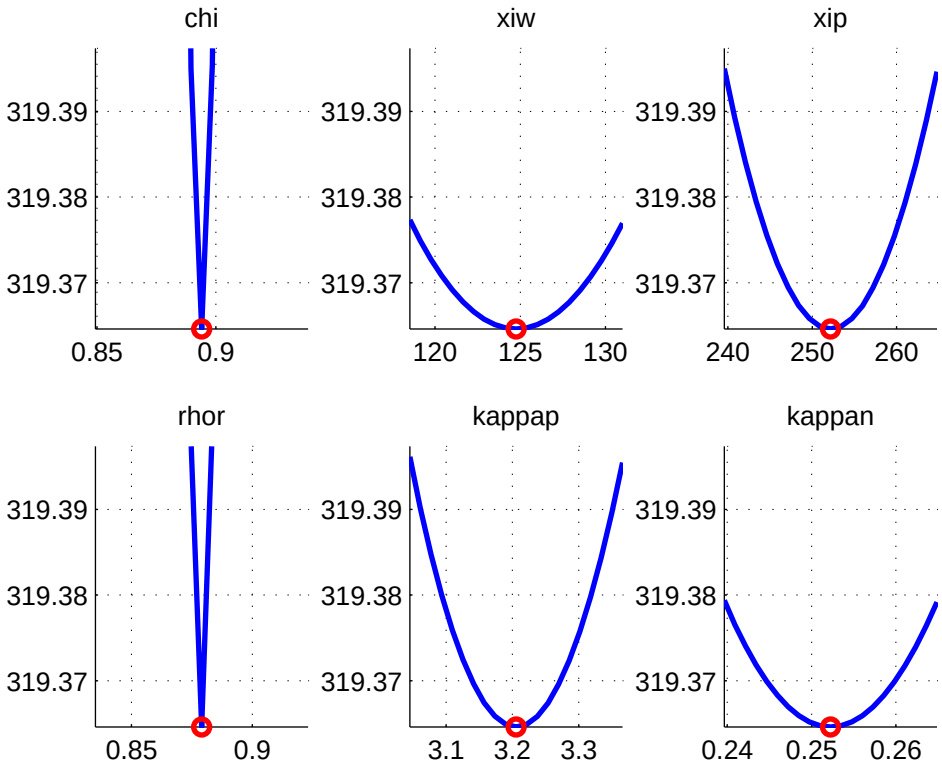


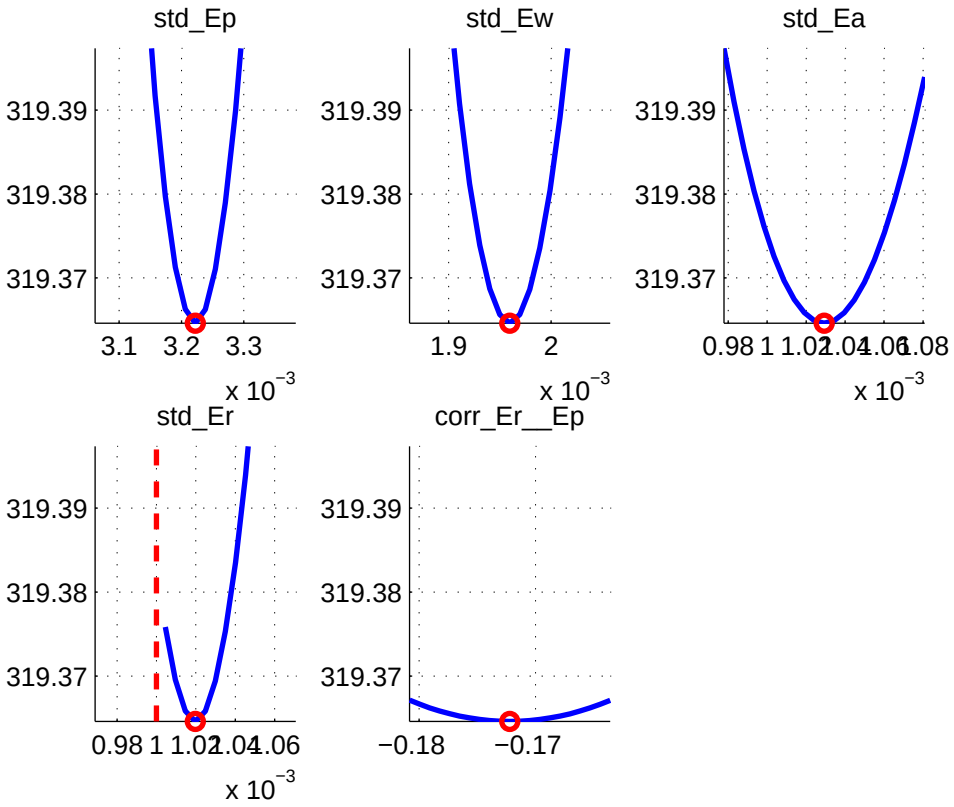


Adjust the style of the graphs:

- `objh` is a vector of handles to the objective function curves;
- `esth` is a vector of handles to the marks depicting the actual estimates;
- `bh` is a vector of handles to the lower and upper bound lines (plotted only if they fall within the graph range).

```
192 set(objh,'lineWidth',2);
193 set(esth,'marker','o','lineWidth',2);
194 set(bh,'lineStyle','--','lineWidth',2);
```


chi
xiw
xip
rhor
kappap
kappan
319.39
319.38
319.37
0.85
0.9
120
125
130
240
250
260
3.1
3.2
3.3
0.24
0.25
0.26



10 Run Metropolis posterior simulator

We run a thousand 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). Then, we calculate some statistics of the simulated parameter chains.

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.

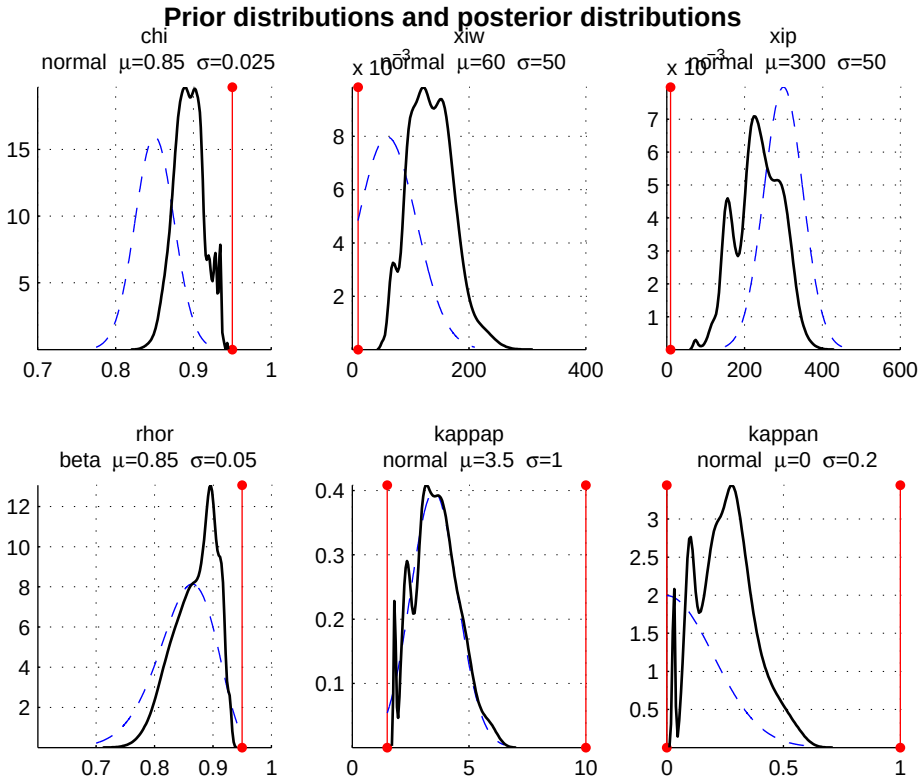
```
209 N = 1000;
210
211 [theta,logpost,ar] = arwm(pos,N, ...
212     'progress',true,'adaptScale',2,'adaptProposalCov',1,'burnin',0.20);
213
214 ar(end)
215
216 s = stats(pos,theta,logpost)
```

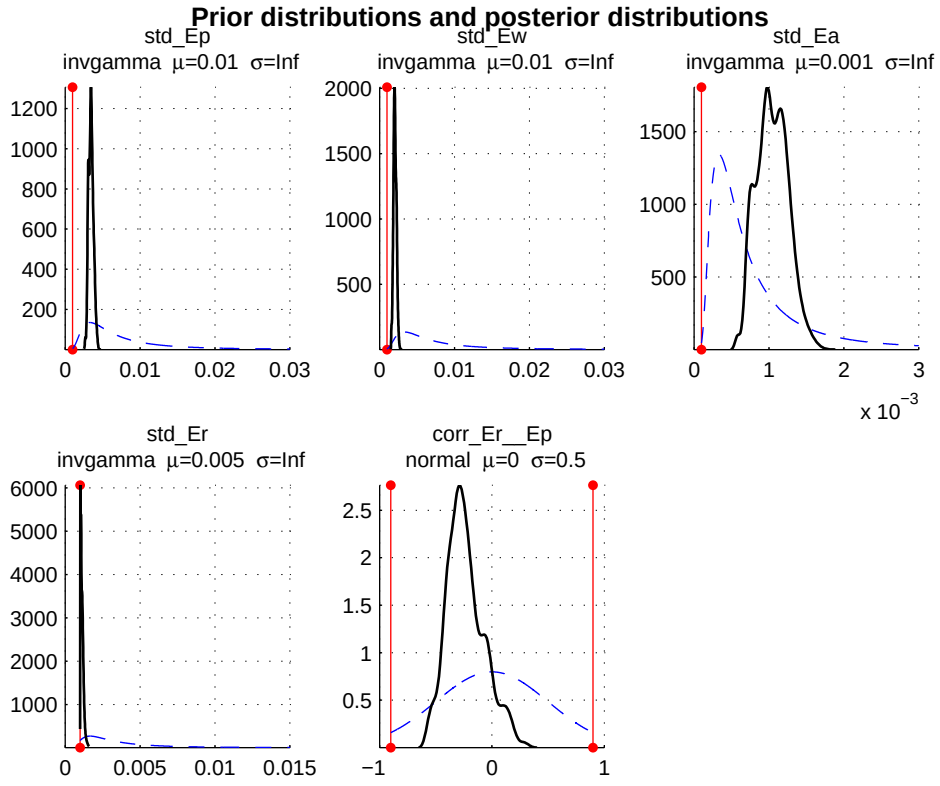
```
[--IRIS poster.arwm progress-------------]
[****************************************]
ans =
    0.2230
s =
      mdd: -349.5146
    chain: [1x1 struct]
     mean: [1x1 struct]
      std: [1x1 struct]
     hpdi: [1x1 struct]
```

11 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 you generate another posterior chain.

```
224 [pr,po,fig,ax,prlin,polin,blin,tit] = plotpp(E,theta,'subplot',[2,3]);
225
226 ftitle(fig,'Prior distributions and posterior distributions');
227
228 set(prlin,'lineStyle','--');
229 set(polin,'color','black','linewidth',1);
230 set(blin,'marker','.','markerSize',11);
231 set([ax,tit],'fontSize',8);
```


Prior distributions and posterior distributions
chi
normal μ=0.85 σ=0.025
xiw
normal μ=60 σ=50
x 10^-3
xip
normal μ=300 σ=50
x 10^-3
rhor
beta μ=0.85 σ=0.05
kappap
normal μ=3.5 σ=1
kappan
normal μ=0 σ=0.2



12 Save model object with estimated parameters

```
235 save('estimate_params.mat','mest');
```

13 Help on IRIS functions used in this m-file

Use either `help` to display help in the command window, or `idoc` to display help in a HTML browser window.

```
help model/estimate
help model/neighbourhood
help poster/arwm
help poster/stats
help grfun/plotpp
help logprior
help logprior.normal
help logprior.lognormal
help logprior.beta
help logprior.gamma
```

```
help logprior.invgamma
help logprior.uniform
```