SC42025 Filtering & Identification

MATLAB EXERCISE HW1

Instructions: Fill in the live script with your code and answers. Then export the live script as a pdf (in the 'Live Editor tab', click on the arrow under 'Save' and then 'Export to pdf').

You are given measurements $y \in \mathbb{R}^{100\times 1}$. Assuming the measurement model

$$y = F\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, R),$$

with $F \in \mathbb{R}^{100\times 2}$ and $R = \sigma_y^2 I$, $\sigma_y = 0.5$, you would like to find an estimate for $[x_1 \;\; x_2]^\top$. You also have a prior given by

$$\theta \sim \mathcal{N}\left([1.1 \;\; 1.1]^\top, \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}\right).$$

Note that this is a difficult estimation problem, since some measurements have a small signal-to-noise ratio, in other words, F is different in each row.

Exercise 1: Batch estimation

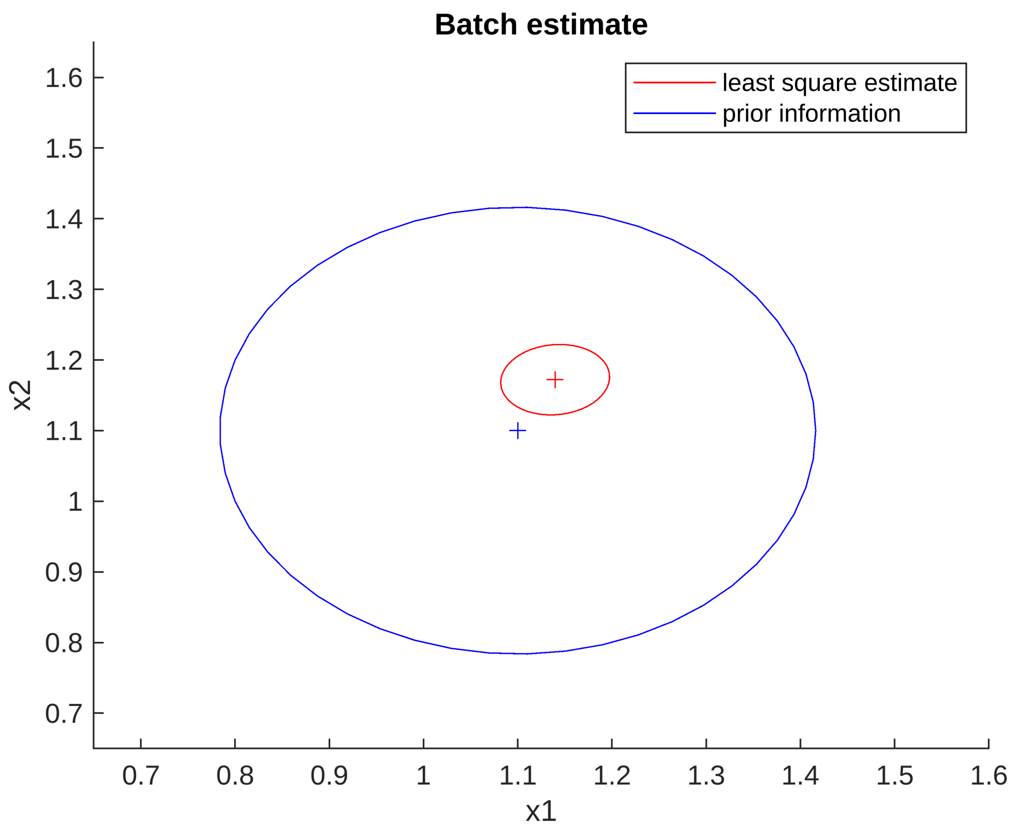
Compute the stochastic least squares estimate using the given matrices `y.mat` and `F.mat` provided in `data.mat`. Print the result.

```
clear; clc; close all;

load('data1.mat')
N       = 100;
invP0   = 10*eye(2);
x0      = [1.1; 1.1];
sigy    = .5;
R       = sigy^2*eye(size(F,1));
Q       = 0.1.*eye(2,2);
mean_e  = x0+Q*F.'*inv(F*Q*F.'+R)*(y-F*x0);
var_e   = Q-Q*F.'*inv(F*Q*F.'+R)*F*Q;

figure
h1=plot_gaussian_ellipsoid(mean_e, var_e);
set(h1,'color','r');
hold on
h1 = plot_gaussian_ellipsoid(x0, Q);
set(h1,'color','b');
hold on
plot(mean_e(1,1),mean_e(2,1),'r+')
hold on
```

```
plot(x0(1,1),x0(2,1),'b+');
xlabel('x1');ylabel('x2');title('Batch estimate');
legend('least square estimate','prior information')
axis([0.65 1.6 0.65 1.65]);
```

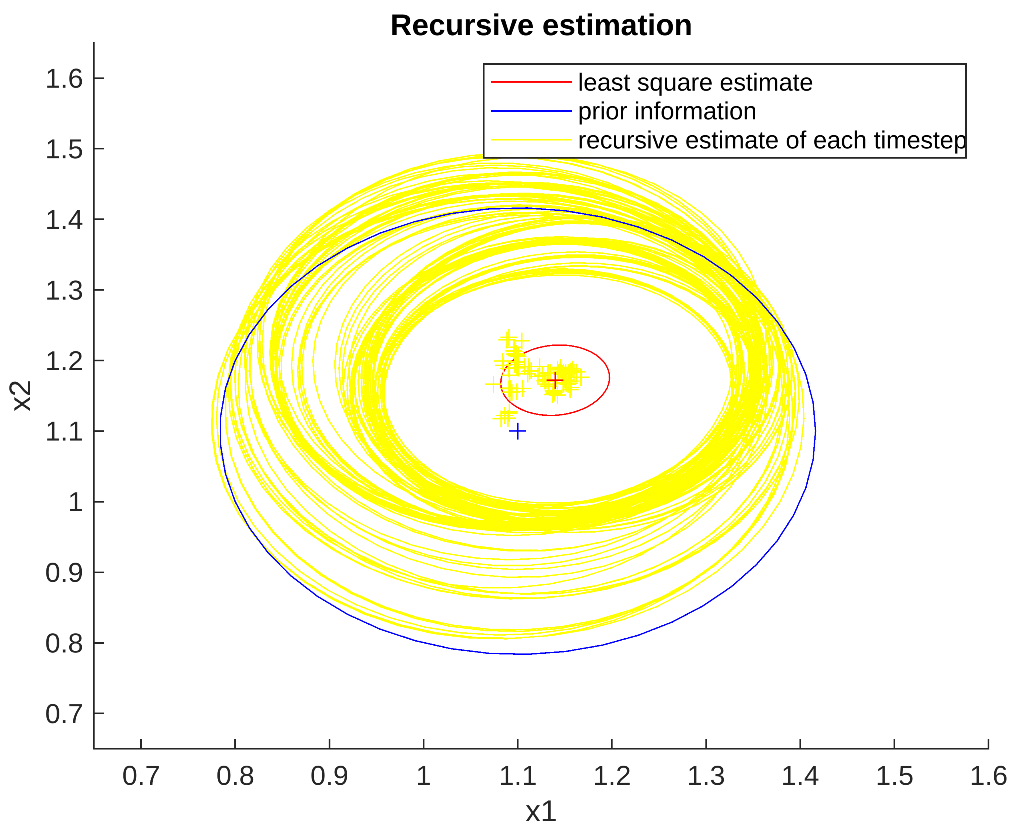


Exercise 2: Recursive estimation

Compute the recursive least squares estimate, pretending that one measurement becomes available at a time. Print the result.

```
mean_v  = zeros(2,99);
w_i     = 0.25;
mean_i  = x0;
var_i   = inv(Q);
varN    = zeros(2,2*99);
for i=1:99
R       = sigy^2*eye(size(i,1));
Q       = 0.1.*eye(2,2);
k_i     = inv(var_i+F(i,:).'*w_i*F(i,:))*F(i,:).'*w_i;
mean_ai = (eye(2,2)-k_i*F(i,:))*mean_i+k_i*y(i,:);
var_aI  = var_i+F(i,:).'*w_i*F(i,:);
mean_i  = mean_ai;
var_i   = var_aI;
m       = 2*i-1;
varN(:,m:m+1)    = inv(var_aI);
mean_v(:,i)      = mean_ai;
```

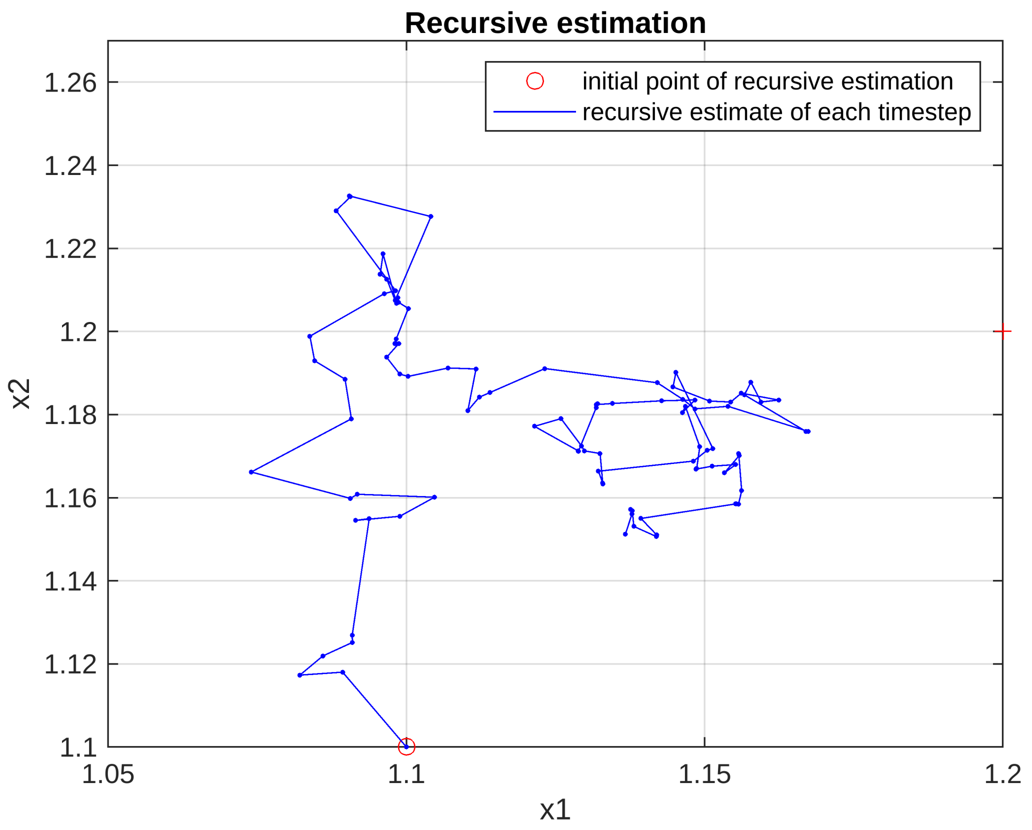
```
end
figure
h1 = plot_gaussian_ellipsoid(mean_e, var_e);
set(h1,'color','r');
hold on
h1 = plot_gaussian_ellipsoid(x0, Q);
set(h1,'color','b');
hold on
for i=1:99
  m       = 2*i-1;
  h1      = plot_gaussian_ellipsoid(mean_v(:,i), varN(:,m:m+1));
  set(h1,'color','y');
end
hold on
plot(mean_v(1,:),mean_v(2,:),'y+')
hold on
plot(x0(1,1),x0(2,1),'b+');
hold on
plot(mean_e(1,1),mean_e(2,1),'r+')
hold on
h1 = plot_gaussian_ellipsoid(x0, Q);
set(h1,'color','b');
xlabel('x1');ylabel('x2');title('Recursive estimation');
legend('least square estimate','prior information','recursive estimate of each timestep
axis([0.65 1.6 0.65 1.65]);
```



Exercise 3: Visualization

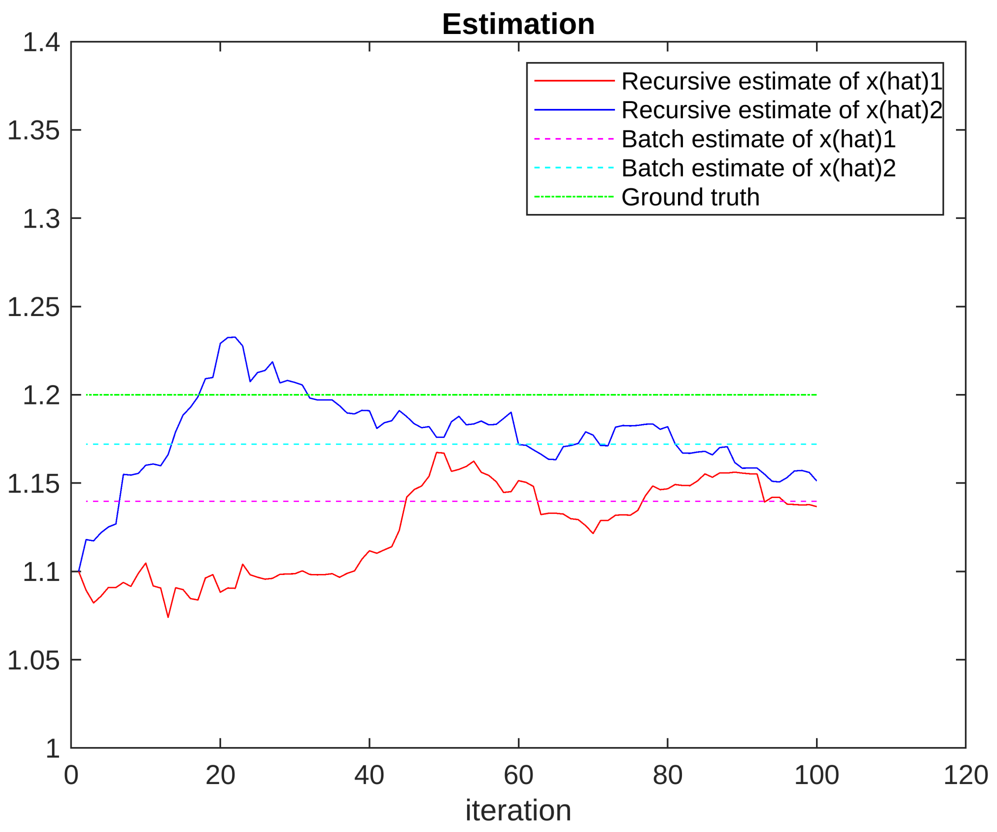
Plot the results of a) and b) **into the same figure**. For b) plot the path connecting the estimates for each recursion step. Note that the recursive estimation should obtain the exact same result as the batch estimation. Also plot the ground truth, which is given by $[1.2 \quad 1.2]$.

```
mean_new=[x0 mean_v];
figure
plot(x0(1,1),x0(2,1),'ro');
hold on
plot(mean_new(1,:),mean_new(2,:),'b-',mean_new(1,:),mean_new(2,:),'b.');grid on;
hold on
plot(1.2,1.2,'r+');%gound truth
xlabel('x1');ylabel('x2');
title('Recursive estimation');
legend('initial point of recursive estimation','recursive estimate of each timestep');
axis([1.05 1.2 1.1 1.27]);
```



```
x=1:100;
figure
plot(x,mean_new(1,:),'r-');
hold on
plot(x,mean_new(2,:),'b-');
hold on
line([100,2],[mean_e(1,1),mean_e(1,1)],'linestyle','--','color','m')
```

```
hold on
line([100,2],[mean_e(2,1),mean_e(2,1)],'linestyle','--','color','c')
hold on
line([100,2],[1.2,1.2],'linestyle','-.','color','g')
hold on
xlabel('iteration');title('Estimation');
legend('Recursive estimate of x(hat)1','Recursive estimate of x(hat)2','Batch estimate
axis([0 120 1 1.4]);
```



```
function h = plot_gaussian_ellipsoid(m, C, sdwidth, npts, axh)

if ~exist('sdwidth', 'var'), sdwidth = 1; end
if ~exist('npts', 'var'), npts = []; end
if ~exist('axh', 'var'), axh = gca; end

if numel(m) ~= length(m)
    error('M must be a vector');
end
if ~( all(numel(m) == size(C)) )
    error('Dimensionality of M and C must match');
end
if ~(isscalar(axh) && ishandle(axh) && strcmp(get(axh,'type'), 'axes'))
    error('Invalid axes handle');
end
```

```
set(axh, 'nextplot', 'add');

switch numel(m)
   case 2, h=show2d(m(:),C,sdwidth,npts,axh);
   case 3, h=show3d(m(:),C,sdwidth,npts,axh);
   otherwise
      error('Unsupported dimensionality');
end

if nargout==0
    clear h;
end
end

%-----------------------------
function h = show2d(means, C, sdwidth, npts, axh)
if isempty(npts), npts=50; end
% plot the gaussian fits
tt=linspace(0,2*pi,npts)';
x = cos(tt); y=sin(tt);
ap = [x(:) y(:)]';
[v,d]=eig(C);
d = sdwidth * sqrt(d); % convert variance to sdwidth*sd
bp = (v*d*ap) + repmat(means, 1, size(ap,2));
h = plot(bp(1,:), bp(2,:), '-', 'parent', axh);
end
%-----------------------------
function h = show3d(means, C, sdwidth, npts, axh)
if isempty(npts), npts=20; end
[x,y,z] = sphere(npts);
ap = [x(:) y(:) z(:)]';
[v,d]=eig(C);
if any(d(:) < 0)
   fprintf('warning: negative eigenvalues\n');
   d = max(d,0);
end
d = sdwidth * sqrt(d); % convert variance to sdwidth*sd
bp = (v*d*ap) + repmat(means, 1, size(ap,2));
xp = reshape(bp(1,:), size(x));
yp = reshape(bp(2,:), size(y));
zp = reshape(bp(3,:), size(z));
h = surf(axh, xp,yp,zp);
end
```