Devon Quaternik

Table of Contents

ELEN 431 HW 4

Problem 1 (taken from code on Camino)

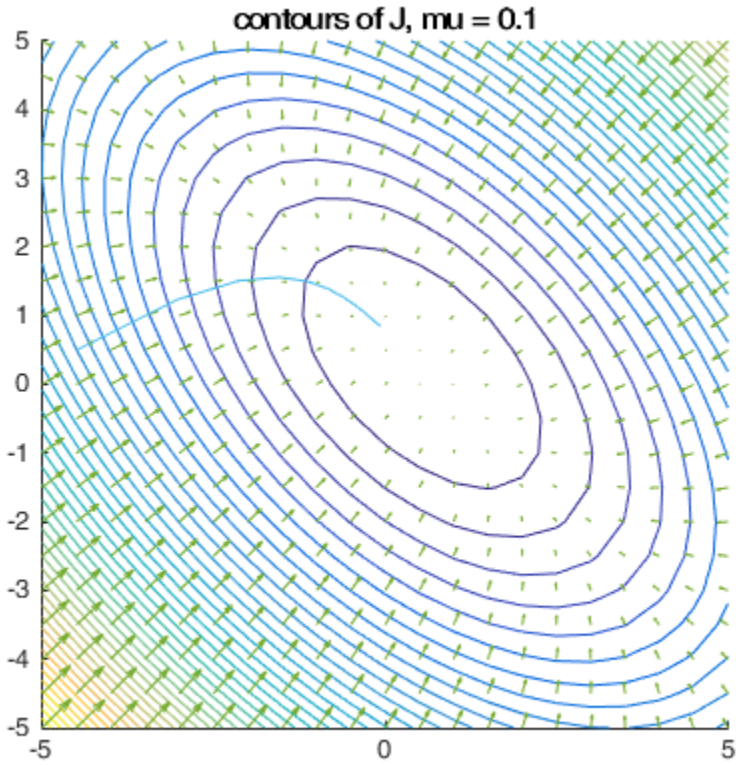
Part 1

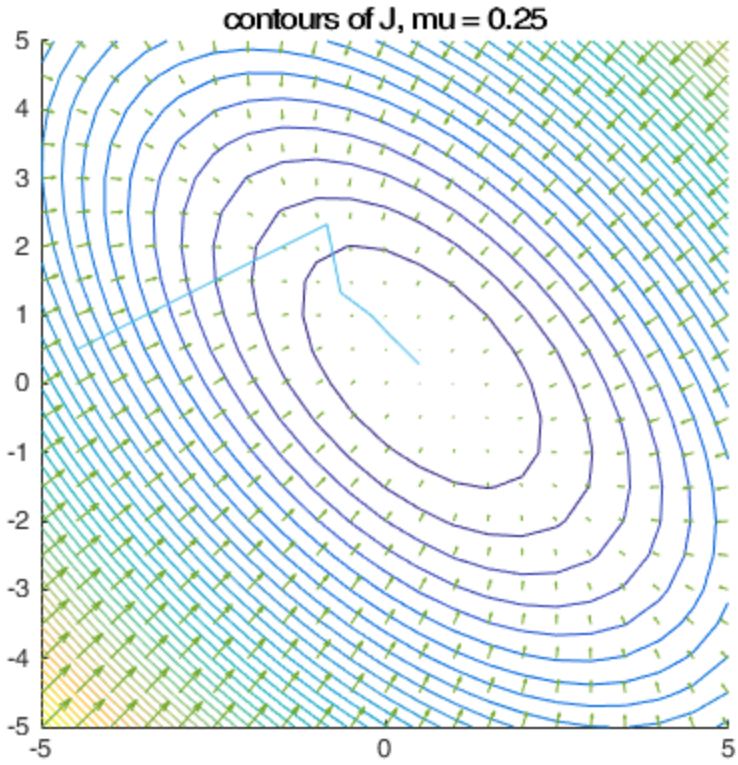
```
% Ru0, Ru1 are the auto-correlation matrix diagonal and corner terms
% P0, P1 are the cross-correlation vector values
% D2 is the power of the desired signal d(n)
% Use grid for display

Ru0 = 1.49;
Ru1 = 0.8;
P0 = 1;
P1 = 0.8;
D2 = 0.1;

%Plot contours of cost function, quiver plot of gradient
figure;
xord=[-5:0.5:5];
yord=[-5:0.5:5];
[x,y]=meshgrid(xord,yord);

%%Quadratic form cost function
J = D2 + Ru0.*x.*x + Ru0.*y.*y + Ru1.*x.*y + Ru1.*y.*x - 2*P0.*x -
 2*P1.*y;

%Gradient computation, reverse sign to get negative gradient (point
 down)
[px,py]=gradient(J,0.5,0.5);
px=-px;
py=-py;

%Plot contours and gradient arrows
figure(1);
hold on
axis('square');
contour(x,y,J,40);
quiver(x,y,px,py);
title('contours of J, mu = 0.1')

figure(2);
hold on
```

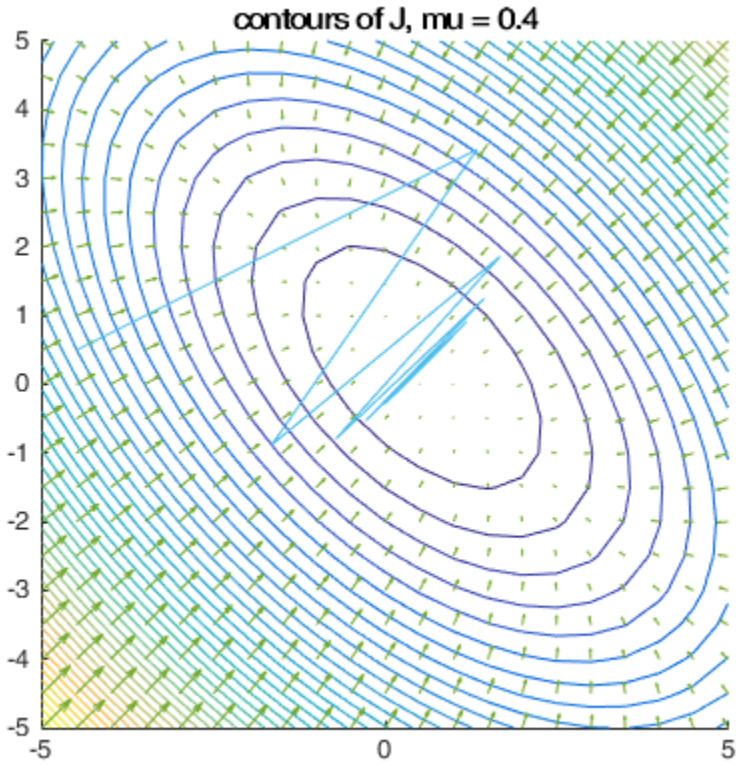
```
axis('square');
contour(x,y,J,40);
quiver(x,y,px,py);
title('contours of J, mu = 0.25')

figure(3);
hold on
axis('square');
contour(x,y,J,40);
quiver(x,y,px,py);
title('contours of J, mu = 0.4')

figure(4);
hold on
axis('square');
contour(x,y,J,40);
quiver(x,y,px,py);
title('contours of J, mu = 0.5')

figure(5);
hold on
axis('square');
contour(x,y,J,40);
quiver(x,y,px,py);
title('contours of J mu = 0.75')


% Set up for steepest descent recursion

Ru = [Ru0 Ru1; Ru1 Ru0];
P = [P0 P1]';
wthold = zeros(11,5);
Jhold = zeros(11,5);
mulist = [0.1 0.25 0.4 0.5 0.75];

%
% Outer loop to cycle through all five values of mu
%
for m = 1:5,
 xa = -4.5;  % initialize weight vector to [-4.5, 0.5];
 ya = 0.5;
 wt1 = [xa];
 wt2 = [ya];
 W = [xa ya]';
 mu = mulist(m);   %Current stepsize
 wt1x = [xa];

 J = -2*W'*P + W'*Ru*W + D2;
 Jt = [J];
%
% Inner loop for 10 steps of recursion
%
 for k = 1:10
  W = W + 2*mu.*(P-Ru*W);
```

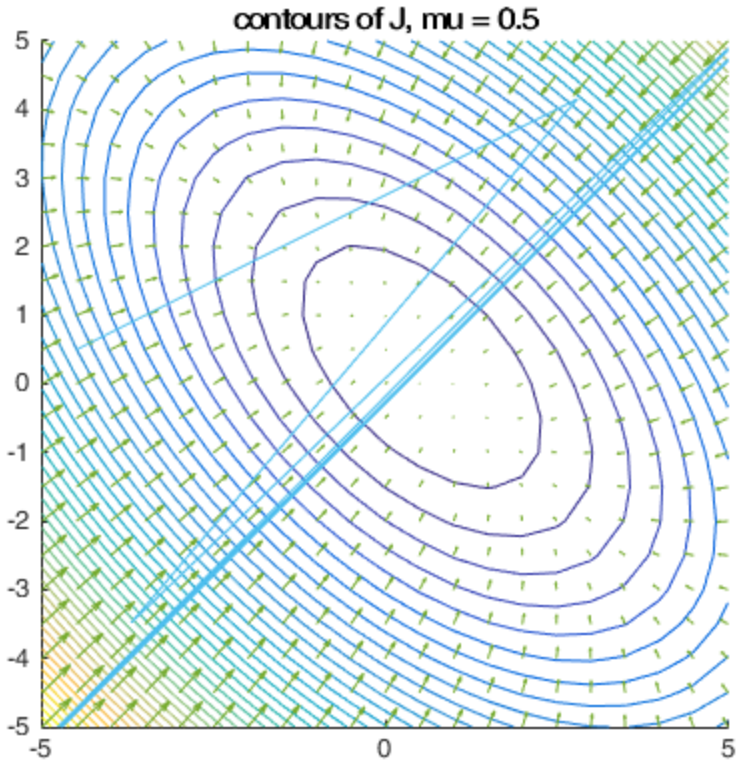
```
  xa = W(1);
  ya = W(2);
  wt1 = [wt1;xa];
  wt2 = [wt2;ya];
  wt1x = [wt1x;xa];
%  J = -2*W'*P + W'*Ru*W + 1;
  J = -2*W'*P + W'*Ru*W + D2;
  Jt = [Jt; J];
    end
    figure(m)
 plot(wt1, wt2);
 wthold(:,m) = wt1x;
 Jhold(:,m) = Jt;
    axis( [-5 5  -5 5])
end



figure
t = [0:1:10];
hold on
for k=1:5,
 plot(t,wthold(:,6-k))
end
legend('.75','0.5','0.4','0.25','0.1');
tx = [0:0.1:10];
ya = 0.516 * ones(1,101);
plot(tx,ya,'.');
%axis( [0,10,-5,5]);
axis( [0,10,-20,20]);
title('Behavior of weight values, dotted is optimal')

figure
hold on
for k=1:5
 plot(t,Jhold(:,6-k))
end
axis( [0,10,0,80] );
title('Learning curve for varying weight values');
legend('.75','0.5','0.4','0.25','0.1');

disp('The max of mu should be 0.5, non-inclusive based on results
 above.')

vals = eig(Ru);
mubound=2/(max(vals))
```

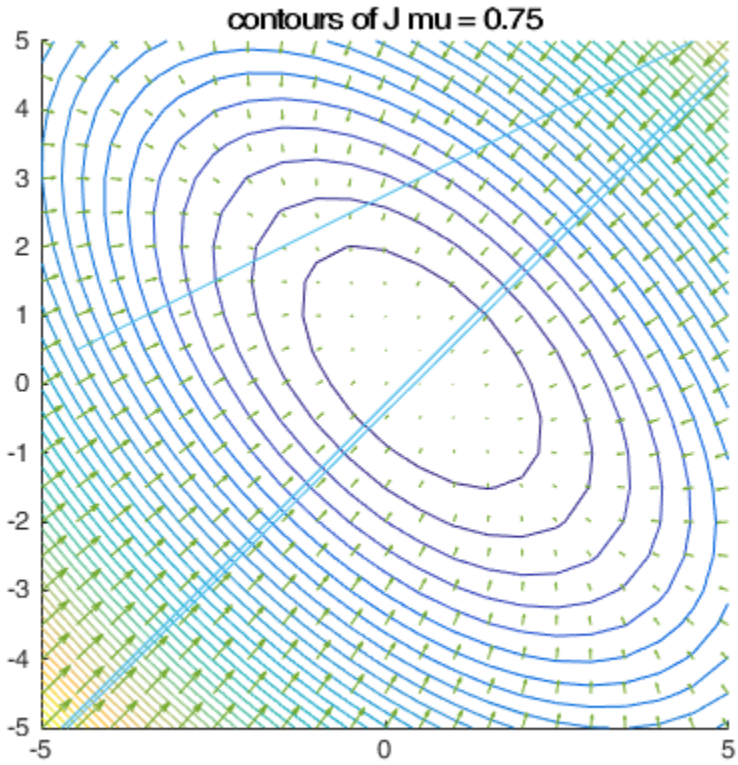
```
The max of mu should be 0.5, non-inclusive based on results above.

mubound =

    0.8734
```

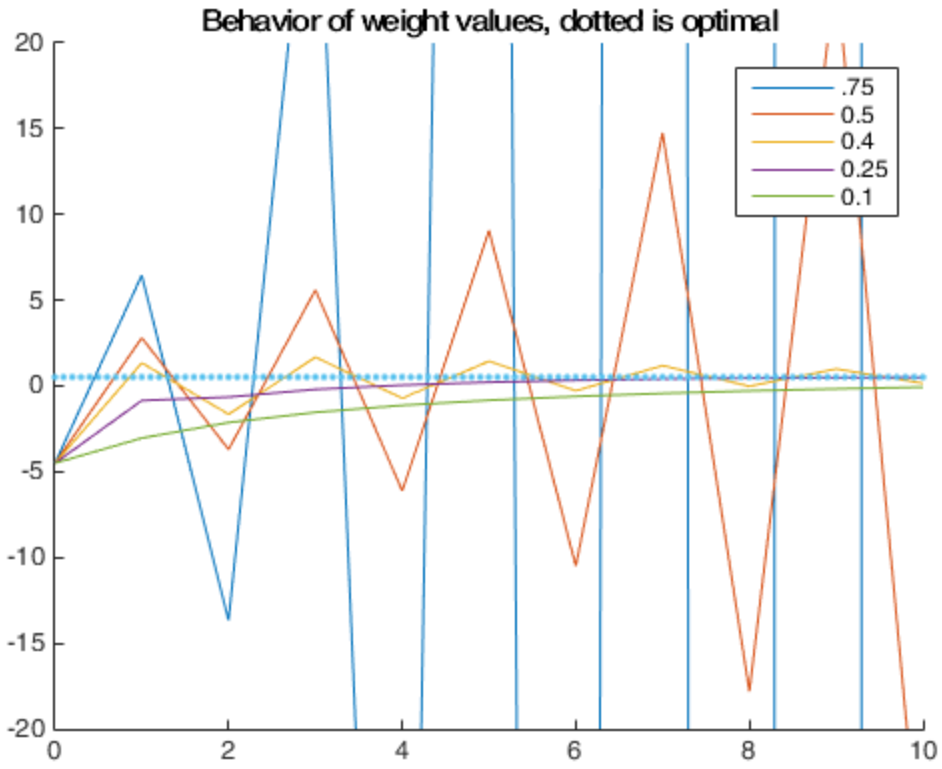

contours of J, mu = 0.1
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
0
5

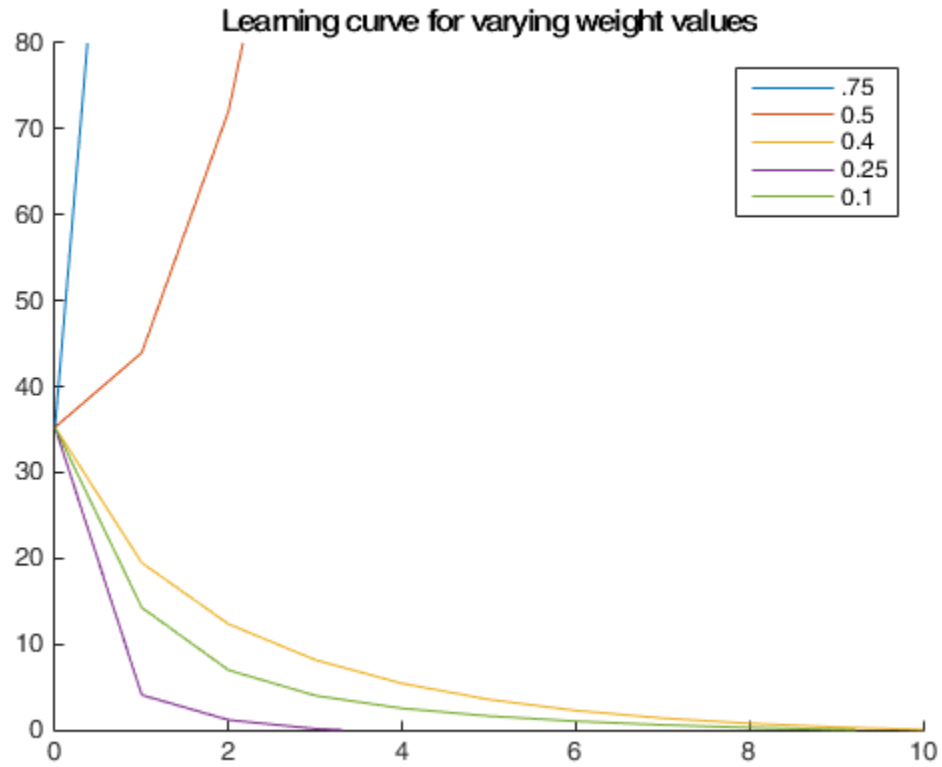

contours of J, mu = 0.25
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
0
5


contours of J, mu = 0.4
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
0
5


contours of J, mu = 0.5
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
0
5

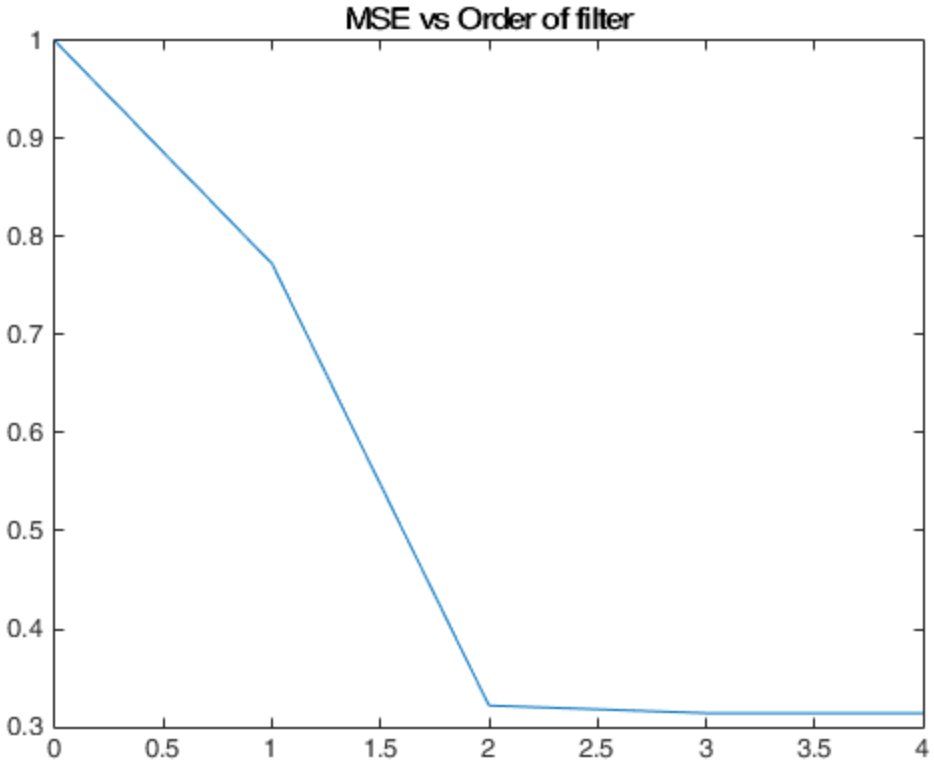

contours of J mu = 0.75
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
0
5


Behavior of weight values, dotted is optimal
.75
0.5
0.4
0.25
0.1
20
15
10
5
0
-5
-10
-15
-20
0
2
4
6
8
10


Learning curve for varying weight values
.75
0.5
0.4
0.25
0.1
80
70
60
50
40
30
20
10
0
0
2
4
6
8
10

Problem 2

```
R4 = [1.1 0.5 0.1 -0.1; 0.5 1.1 0.5 0.1; 0.1 0.5 1.1 0.5; -0.1 0.1 0.5
 1.1];
p4 = [0.5; -0.4; -0.2; -0.1];
sigd2 = 1.0;
sigv2 = 0.1;

%MSE for Weiner filter with length 0,1,2,3,4
%Jmin = sigd2 - ph*wopt = sigd2 -ph*inv(R)*p

for M = 0:4
    if M < 5 && M > 0
        R = R4(1:M,1:M);
        R1 = inv(R);
        p = p4(1:M);
    else
        R = 0;
        R1 = 0;
        p = 0;
    end
    Jmin(M+1) = sigd2 - p'*R1*p;
end

Jmin

figure;
plot([0:M],Jmin)
title('MSE vs Order of filter')


Jmin =

    1.0000    0.7727    0.3219    0.3141    0.3141
```



Published with MATLAB® R2015a