Q8

close all

clear all

p\_mean = [1.1 1.35 1.25 1.05];

% sigma = [0.2 -0.2 -0.12 0.02;

% -0.2 1.4 0.02 0;

% -0.12 0.02 1 -0.4;

% 0.02 0 -0.4 0.02];

sigma = [0.2 -0.2 -0.12 0.02;

-0.2 1.4 0.02 0;

-0.12 0.02 1 -0.4;

0.02 0 -0.4 0.02];

H = sigma;

f = [0;0;0;0];

A = [-1 0 0 0; % -x1 <= 0

0 -1 0 0; % -x2 <= 0

0 0 -1 0; % -x3 <= 0

0 0 0 -1; % -x4 <= 0

-1.\*p\_mean]; % trans(-p\_mean)x <= -r\_min

Aeq = [1 1 1 1];

beq = 1;

for i = 1:1:15

b(:,i) = [0;0;0;0;-0.1\*i]; % -r\_min = 0.1\*i

[x(:,i), fval(:,i), exitflag(:,i), output, lambda] = ...

quadprog(H, f, A, b(:,i), Aeq, beq);

end

% x, fval, exitflag

exp\_return = p\_mean \* x;

figure(1)

p1 = plot(exp\_return, fval);

hold on

p2 = plot([1.3 1.3], [-0.5 1], 'r');

set(gca,'linewidth',2)

set(p1, 'linewidth', 3)

set(p2, 'linewidth', 3)

xlabel('expected return')

ylabel('risk of profolio')

legend('expected return vs risk of profolio')

grid

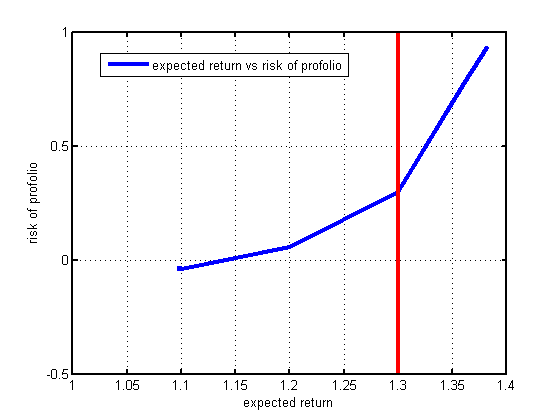
hold off

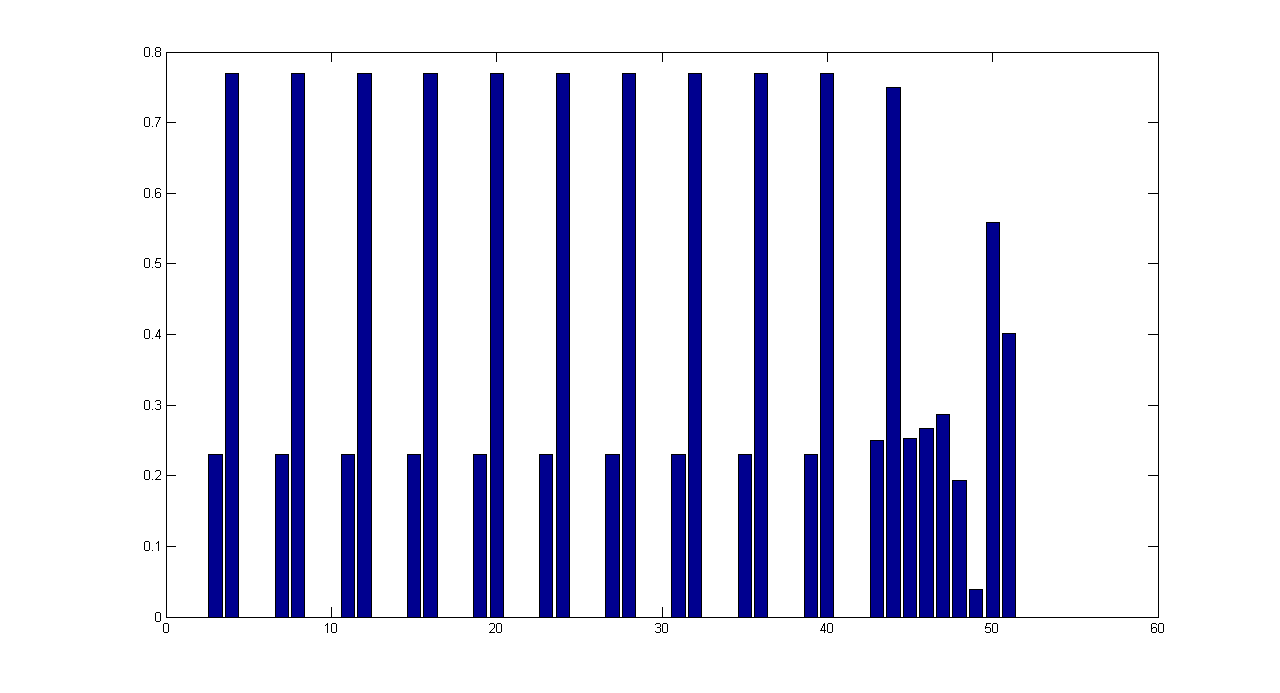
figure(2)

bar([x(:,1); x(:,2); x(:,3); x(:,4); x(:,5); x(:,6); x(:,7); x(:,8); x(:,9);

x(:,10); x(:,11); x(:,12); x(:,13)])

(To the right of the line indicates the feasible region)





Minimum expected return constraint: 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3

Composition

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6.39461435480326e-17 | 6.86670287399952e-17 | 7.38121847943036e-17 | 2.04573676885185e-16 | 1.01268802778177e-15 | 3.00235206255833e-15 | 3.81713767141727e-15 | 2.26539303203320e-15 | 3.54037352048975e-16 | 2.11106650101455e-16 | 1.73029137194440e-16 | 0.253179811359817 | 0.0391534391523836 |
| 5.17034233759066e-13 | 5.55124395683706e-13 | 5.96628176618747e-13 | 1.65126769815236e-12 | 8.16415285934602e-12 | 2.42312344234802e-11 | 3.08594160357170e-11 | 1.83744871734884e-11 | 2.85021838961996e-12 | 1.65587089531608e-12 | 8.94064451650595e-17 | 0.266871523709986 | 0.558730158730200 |
| 0.230769230769106 | 0.230769230769097 | 0.230769230769087 | 0.230769230768832 | 0.230769230767257 | 0.230769230763375 | 0.230769230761773 | 0.230769230764792 | 0.230769230768542 | 0.230769230768846 | 0.250000000000958 | 0.286397761610403 | 0.402116402116620 |
| 0.769230769230377 | 0.769230769230348 | 0.769230769230317 | 0.769230769229517 | 0.769230769224577 | 0.769230769212391 | 0.769230769207363 | 0.769230769216832 | 0.769230769228607 | 0.769230769229498 | 0.749999999999042 | 0.193550903319795 | 7.96954365497068e-13 |

Comments: the diversity of the composition will give a better expected return but also keep small risk.

Q9

We need to relate the force applied to the unit mass object and its velocity and position, defined by and , respectively in the below Matlab code. The displacement is defined by . We know that within each time instance (t=1 in this case, the modal is discrete in time instance.)

Where , And from the constraint in a)

See the below table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| f |  |  |  |  | … | … | … | … | … |  |
| v |  |  |  |  | … | … | … | … | … |  |
| s |  |  |  |  | … | … | … | … | … |  |
| p |  |  |  | … | … | … | … | … | … |  |

Therefore, re-organize and the rewrite the constraint

Therefore

For b), just add one more constraint:

close all

clear all

H = ones(10,10);

%%%%constraint for a, uncomment this%%%%%%

%Aeq = [1 1 1 1 1 1 1 1 1 1;9.5 8.5 7.5 6.5 5.5 4.5 3.5 2.5 1.5 0.5];

%beq = [0;1];

%%%%constraint for a, uncomment this%%%%%%

Aeq = [1 1 1 1 1 1 1 1 1 1;9.5 8.5 7.5 6.5 5.5 4.5 3.5 2.5 1.5 0.5;...

4.5 3.5 2.5 1.5 0.5 0 0 0 0 0]; % constraint for b

beq = [0;1;0]; % constraint for b

[x,fval,exitflag,output,lambda] = ...

quadprog(H,[],[],[],Aeq,beq);

% x = f, force

v = zeros(10,1); % velocity

for t = 1:1:10

v(t,:) = sum(x(1:t));

end

a = [0;x]; % add a zero to the front, acceleration

s = zeros(10,1); % displacement

for t = 1:1:10

s(t,:) = sum(a(1:t)) + 0.5 \* x(t,:);

end

p = zeros(10,1); % position

for t = 1:1:10

p(t,:) = sum(s(1:t));

end

figure(1)

p1 = plot(x);

set(gca,'linewidth',2)

set(p1, 'linewidth',3)

xlabel('time')

ylabel('force')

legend('force')

grid on

figure(2)

p1 = plot(v);

set(gca,'linewidth',2)

set(p1, 'linewidth',3)

xlabel('time')

ylabel('velocity')

legend('velocity')

grid on

figure(3)

p1 = plot(p);

set(gca,'linewidth',2)

set(p1, 'linewidth',3)

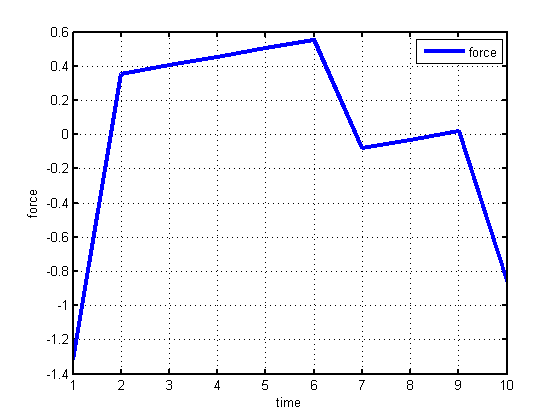
xlabel('time')

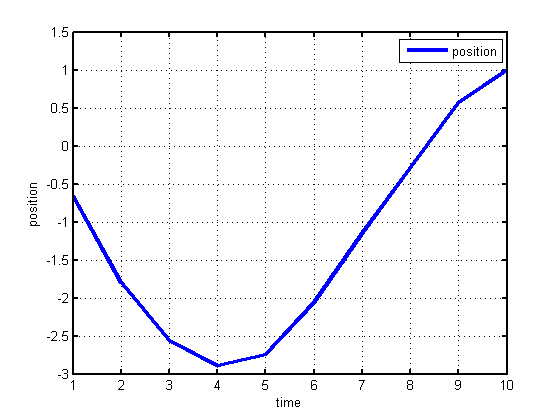
ylabel('position')

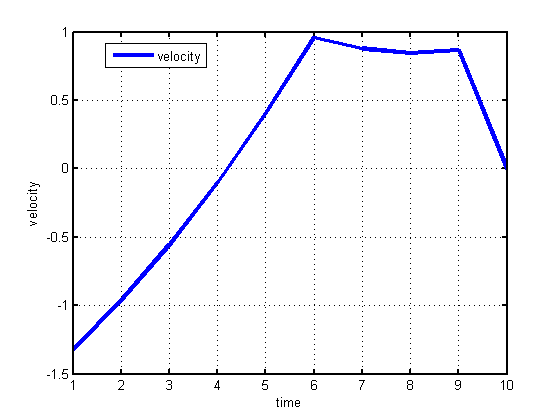
legend('position')

grid on

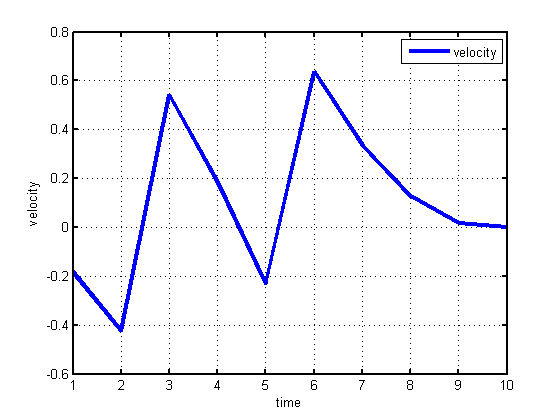
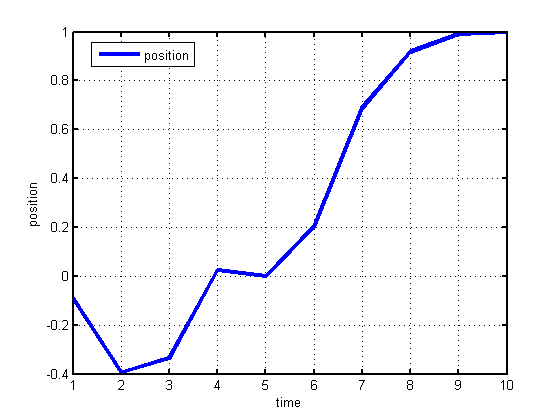
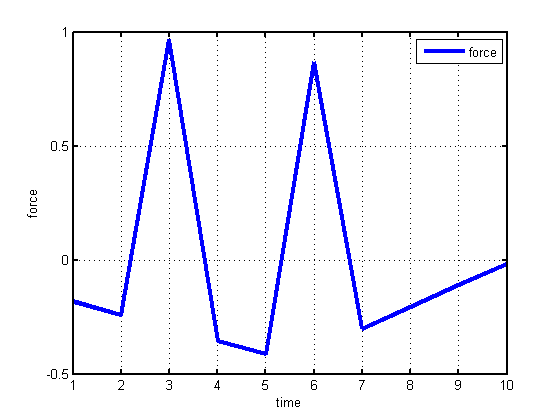
Part a)







Part b)



Q10

Reformulating the least-square optimization problem; since

Is effectively the same as

Or one more step

Therefore, we need to rewrite so that it is a standard form quadratic programming (Q.P.) optimization problem. Recall the circulant matrix, which is used to write the convolution of two vectors in linear product fashion such that

Define circulant matrix C, where each row will be the shifted version of impulse response vector h(n).

Where means the row of C, where , meaning that after the channel output will be zero, also means there are in total 29 time steps. The dimension of C is related to the dimension of g(m) and h(n):

Which means C has 29 rows and 20 columns. The matrix C is computed through Matlab code below. After the calculation of C, can be rewritten as

A is the product of , which is the coefficient (square) matrix of each time instance. This is defined below in Matlab code. Further rewrite as

P is the summation of all A, also defined in the below Matlab code. Rewrite the optimization problem as

It is a Q.P.

clear all

load('hw2data.mat','h')

load('hw2data.mat','y')

m = 20; % dimension of g

n = 10; % dimension of h

C = zeros(m+n-1,m) % initialize circulant matrix

H = transpose([transpose(h) zeros(1,10)]); % impulse response with zero padding

B = zeros(size(H)); % buffer

%% Formulating Circulating Matrix

for i=-(n+(n-1)):1:-1

B(-i+1:end) = H(1:end+i);

C(i+m,:) = fliplr(transpose(B));

end

for q=m:1:(m+n-1)

k = 1; % right shifting each row by one

B = C(q-1,:);

B(k+1:end)=B(1:end-k);

C(q,:) = B;

end

%% Formulating the coefficient for Q.P., defined by matrix P

A = zeros(m,m,m+n-1);

for j=1:1:(m+n-1) % in total there are 29 time samples are meaningful

A(:,:,j) = transpose(C(j,:))\*(C(j,:)); % second-order term coefficicent for each time step

end

P = zeros(m:m);

for k=1:1:(m+n-1)

P = P + A(:,:,k); % pull out the variable g, the summation of all second-order term coefficicent for each time step

end

%% Formulating Q.P., with variable D (delay), and variable g (equalizer)

H = P;

f = transpose(zeros(1,m));

objective = (zeros(1,m+n-1));

for D = 1:1:(m+n-1)

Aeq = C(D,:); % the constraint at t = D

beq = 1;

[x(:,D),fval(:,D),exitflag(:,D),output,lambda] = ...

quadprog(H,f,[],[],Aeq,beq);

objective(:,D) = transpose(x(:,D))\*P\*(x(:,D)) - 1; % objective function, {sum(cov(g\*m))^2(t), t not equal to D}

end

%% Plot objective function versus D

D = 1:1:(m+n-1);

figure(1)

p1 = plot(D, objective);

set(gca,'linewidth',2)

set(p1, 'linewidth',3)

xlabel('delay D')

ylabel('sum of squared errors')

legend('delay vs sum of squared errors')

grid on

% D = 9 gives the optimal solution, check C(9,:)\*x(:,9) to see if the

% constraint is satisfied

%% Apply deconvolution to the data

g = x(:,9); % best least-square equalizer

z = conv(y,g);

z = z(n-1:10000+(n-1)-1); % 10000 samples, reconsrtucted

% normalize the data

% for r=1:1:10000

% z(r,:) = z(r,:)/abs(z(r,:));

% end

figure(2)

hist(y)

xlabel('amplitude of y')

ylabel('count')

figure(3)

hist(z)

xlabel('amplitude of z')

ylabel('count')

