Problem 1

```
% Section A

num=xlsread('HW2_Due20180303.xlsx','SPX_DXY_MonthlyReturns');
[text,text,nb] =xlsread('HW2_Due20180303.xlsx','SP500Stocks');
stock = xlsread('HW2_Due20180303.xlsx','SP500Stocks');
stock_mre =stock(2:end,2:end);
n=length(stock_mre(1,1:end))
```

```
n = 210
```

```
sp500_mre= num(1:end,2);
USDI_mre =num(1:end ,3);
snum = length(stock_mre(:,1))
```

```
snum = 505
```

```
%sectors name
sec_name=unique(text(3:end,3))
```

```
sec_name = 11×1 cell 数组
    'DSCR'
    'ENER'
    'FINA'
    'HLTH'
    'INDU'
    'INFT'
    'MATS'
    'REAL'
    'STPL'
    'TCOM'
    'UTIL'
```

```
%market exposure matrix
beta_mat=nan(2,snum);

val_mat= [sp500_mre,USDI_mre]
```

```
val_mat =
   -0.9844   -1.9168
   -6.2602   -3.2087
   -8.0752   -0.0088
    1.9069    1.2785
    7.6706    1.1057
    0.8760    0.5339
   -1.4593    2.9636
   -1.9285   -0.8735
    3.7609   -0.4532
   -6.0628   -2.8916
```

```
for i=1:n

    y= stock_mre(i,:)';

    %   add the intercept column (which is for the residual return for this
    %   APT model)
    X=[ones(length(y),1),val_mat];
```

```
    %  regression to find the stock's factor exposures over time
    [b,bint,r]=regress(y,X);

    %  record the exposures
    %  note: the first element in b is the residual
    beta_mat(:,i)=b(2:end);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   add sector risk factors (GICS Sectors)

%   the set of unique sectors (abbreviations)
sector_set=sec_name
```

```
sector_set = 11×1 cell 数组
    'DSCR'
    'ENER'
    'FINA'
    'HLTH'
    'INDU'
    'INFT'
    'MATS'
    'REAL'
    'STPL'
    'TCOM'
    'UTIL'
```

```
%number of stocks
num_stock =snum
```

```
num_stock = 505
```

```
%   number of sectors
num_sector=length(sector_set);

%   sector exposure matrix
sector_exposure=zeros(num_sector,num_stock);

%   assign sector number to each stock
[blah,stock_sector]=ismember(text(3:end,3),sector_set);

%   fill in the exposure value (1 or 0)
for i=1:num_stock
   sector_exposure(stock_sector(i),i)=1;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   combine the three factors and the sector factors
[num,factor_list]=xlsread('HW2_Due20180303.xlsx','SPX_DXY_MonthlyReturns');
factor_list=[factor_list';sector_set];

%   append sector exposures to the beta_mat
beta_mat=[beta_mat;sector_exposure];

%   we'll delete the sector exposure column for the last sector(which is the ULT
sector), because its value
%   is redundant given sector exposure information for all other sectors.
factor_list=factor_list(1:end-1);
```

```
beta_mat=beta_mat(1:end-1,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   calculate factor returns at each time point
%   note: based on the second step on Fama-Macbeth regression

%   the total number of factors
num_factor=length(factor_list)+1;

%   pre-allocate memory for the factor return matrix
%   note: each column is for one factor
num_month =n
```

```
num_month = 210
```

```
factor_ret=nan(num_month,num_factor);
stock_mre =stock_mre';


%%size factor
stock_cap=nb(3:end,4);
stock_cap=cell2mat(stock_cap);
stock_cap =log10(stock_cap);
stock_info = text(3:end,3);
stock_info =cell2mat(stock_info);
sector_set =cell2mat(sector_set);
stock_cat = nan(num_sector, num_stock);
count = zeros(1,11);

for i = 1:505
   for j =1:11
      if(~isnan(stock_mre(t,i)))
         if(stock_info(i,1)==sector_set(j))
            stock_cat(j,count(j)+1)=stock_cap(i);
            count(j)=count(j)+1;
         end
      end
   end
end

%%create an array sector_info to store the mean and std of each sector's mean
and std.

sector_info = nan(11,2);
for i =1:11
   sector_info(i,1)= mean(stock_cat(i,1:count(i)));
   sector_info(i,2) = std(stock_cat(i,1:count(i)));
end

%create an array stock_size to show the size of each stock.
for i =1:505
   for j = 1:11
      if(~isnan(stock_mre(t,i)))
         if (stock_info(i,1:4)==sector_set(j,1:4))
            stock_size(t,i) = (stock_cap(i)-
sector_info(j,1))/(sector_info(j,2));
         end
      end
   end
end

%%append stock size into beta-matrix
```

```
beta_mat =[beta_mat;stock_size(end,:)];

for t=1:num_month

    %   the vector of stocks' returns at time t
    %   note: each row has one stock's return

    y=(stock_mre(t,:))';

    %   the factor exposures
    %   note: each row has an array of factors exposures for one stock
    X=beta_mat';

    %   add the intercept column (which is for the residual return for this
    %   APT model)
    X=[ones(length(y),1),X];

    %   run the cross-section regression
    [b,bint,r]=regress(y,X);

    %   record factors' returns at the time poin t
    factor_ret(t,:)=b(2:end);

end




factor_list =[factor_list;'size factor'];









%   now we have a time series of monthly returns for each factor,
%   we can estimate the factor risk premium magnitude for each factor
for k=1:num_factor
    fprintf('--------- %s ------------\n',factor_list{k});

    %   the return vector
    tmp_ret=factor_ret(:,k);

    %   monthly risk premiun estimate based on mean factor returns
    est_rp=nanmean(tmp_ret);

    %   the standard error of this estimate
    se_rp=nanstd(tmp_ret)/sqrt(length(tmp_ret));

    %   the t-stats of the estimate
    tstat_rp=est_rp/se_rp;

    fprintf('Est Monthly RP = %.2f%%.\n',est_rp);
    fprintf('Std Error.=%.2f%%.\n',se_rp);
    fprintf('T-Stat=%.2f.\n\n',tstat_rp);
end
```

```
  --------- S&P - 500 Index ------------
```

```
Est Monthly RP = 0.20%.

Std Error.=0.39%.

T-Stat=0.51.

--------- US Dollar Index ------------

Est Monthly RP = 0.07%.

Std Error.=0.25%.

T-Stat=0.29.

--------- DSCR ------------

Est Monthly RP = 0.55%.

Std Error.=0.39%.

T-Stat=1.43.

--------- ENER ------------

Est Monthly RP = 0.19%.

Std Error.=0.43%.

T-Stat=0.44.

--------- FINA ------------

Est Monthly RP = -0.11%.

Std Error.=0.38%.

T-Stat=-0.29.

--------- HLTH ------------

Est Monthly RP = 0.40%.

Std Error.=0.32%.

T-Stat=1.24.

--------- INDU ------------
```

```
Est Monthly RP = 0.29%.

Std Error.=0.35%.

T-Stat=0.81.

--------- INFT ------------

Est Monthly RP = 0.58%.

Std Error.=0.35%.

T-Stat=1.64.

--------- MATS ------------

Est Monthly RP = 0.44%.

Std Error.=0.37%.

T-Stat=1.19.

--------- REAL ------------

Est Monthly RP = 0.55%.

Std Error.=0.33%.

T-Stat=1.67.

--------- STPL ------------

Est Monthly RP = 0.18%.

Std Error.=0.27%.

T-Stat=0.68.

--------- TCOM ------------

Est Monthly RP = 0.08%.

Std Error.=0.35%.

T-Stat=0.23.

--------- size factor ------------
```

```
Est Monthly RP = 0.03%.

Std Error.=0.05%.

T-Stat=0.53.
```

```
% Section B

%B.
X= beta_mat;
%%aggregate exposure on factors
M=inv(X'*X)*X'
%prove all factor portfolios are long-short neutral
neu=zeros(1,13)

for i =1:13

   neu(i)=sum(M(i,:))

end

neu
neu=
     0  0  0  0  0  0  0  0  0  0  0  0  0
%% example: sum(M(3,:)) =0

%%prove any factor potfolio is unit exposure to its own factor,but zero
exposure to all other factors

for i =1:13
   for j= 1:13

   M(i,:)*X(:,j)

   end
end

%%Sample Outcome

%M(3,:)*X(:,3) =1

%M(3,:)*X(:,6) =0
```

Conclusion:

1. No t-stat value is greater than 2, so the portfolio returns are not dependent on any factors.
2. All factors portfolios are long-short neutral since the weight sum of individul factor portfolio is 0
3. Any factor portfolio is unit exposure to its own factor but zero exposure to all

other factors, which proves any individual factor portfolio does not depend on other factors.

```
%%%Problem 2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
fprintf('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %PROBLEM 2=%.2f%%.\n')
```

```
%%%%%%%%%%%%%%%%%%%
```

```
%%%Problem 2

sec_ret=factor_ret(:,3:12);
sec_name =factor_list(3:12);

%   load monthly return information for asset classes in the CAPM model

ac_name=sec_name;
ac_mret =sec_ret;

%   the number of months
num_month=210;

%   the asset classes loaded
% ac_name=ac_name';

%   number of asset classes
num_ac=length(ac_name);
%find Sigma
Sigma=nancov(ac_mret)*12;
%find Pi
sec_aver =zeros(10,1);
for i =1:10
    sec_aver(i,1)=mean(sec_ret(:,i));
end

Pi=sec_aver;
%optimal portfolio without active view

% annulized volatility of portfolio not greater than 10%

%   objective function
obj_func1=@(w) -w'*Pi;
% There is no linear constraint
A=[];
b=[];
Aeq=[];
beq=[];


%  use the equal-weighted portfolio as starting point
w0=ones(num_ac,1)/num_ac;

%   we need to add a non-linear constraint, which is:
% w'*Sigma*w<=(10/sqrt(12))^2
%   run the optimization
nonlcon=@circlecon;
```

```
[w_opt1,fval]=fmincon(obj_func1,w0,A,b,Aeq,beq,[],[],@(w)
nonlcon(w,Sigma,(10/sqrt(12))^2));
```

```
Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the default value of the optimality tolerance,
and constraints are satisfied to within the default value of the constraint
tolerance.

<stopping criteria details>
```

```
port1 =w_opt1;
% annulized volatility of portfolio not greater than 10%


%% With active view
%  the number of years of observations in the data set
T=num_month/12;

%  set tau to 1/T
%  note: within the value range in method 1.
tau=1/T
```

```
tau = 0.0571
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  locate the ENER
tmp_pos=2;
fprintf('---View 1---\n(Long Leg) %s.\n',ac_name{tmp_pos});
```

```
---View 1---
(Long Leg) ENER.
```

```
%  build the view portfolio corresponding to View 1
num_ac =10;
port_v1=zeros(1,num_ac);
port_v1(tmp_pos)=1
```

```
port_v1 =
     0     1     0     0     0     0     0     0     0     0
```

```
%  the view portfolio's expected return
q1=0.2;

%  compared to equlibrium return of the view portfolio
fprintf('View 1: equlibrium return=%.2f%%, active view
return=%.2f%%.\n',port_v1*Pi,q1)
```

```
View 1: equlibrium return=0.19%, active view return=0.20%.
```

```
%%%%%%%%%%%
%  locate the FINA and INDU SECTORS
tmp_long=3; %  for the long leg
tmp_short=5; %  for the short leg
fprintf('---View 2---\n(Long Leg) %s.\n(Short
```

```
Leg) %s.\n',ac_name{tmp_long},ac_name{tmp_short});
```

```
---View 2---
(Long Leg) FINA.
(Short Leg) INDU.
```

```
%  build the view portfolio corresponding to View 2
port_v2=zeros(1,num_ac);
port_v2(tmp_long)=1;
port_v2(tmp_short)=-1
```

```
port_v2 =
     0     0     1     0    -1     0     0     0     0     0
```

```
%  the view portfolio's expected return
q2=0.05;

%  compared to equlibrium return of the view portfolio
fprintf('View 2: equlibrium return=%.2f%%, active view
return=%.2f%%.\n',port_v2*Pi,q2)
```

```
View 2: equlibrium return=-0.40%, active view return=0.05%.
```

```
%%%%%%%%%%%%%%%%%%
%  locate Sector INFT and Sector HLTH and Sector REAL and Sector STPL
tmp_long=[4 6]; %  for the long leg
tmp_short=[8 9]; %  for the short leg
fprintf('---View 3---\n(Long Leg) 1)%s; 2)%s.\n(Short Leg) 1)%s; 2)%s.\n',...

ac_name{tmp_long(1)},ac_name{tmp_long(2)},ac_name{tmp_short(1)},ac_name{tmp_sho
rt(2)});
```

```
---View 3---
(Long Leg) 1)HLTH; 2)INFT.
(Short Leg) 1)REAL; 2)STPL.
```

```
%  build the view portfolio corresponding to View 3
port_v3_long=zeros(1,num_ac);
port_v3_short=zeros(1,num_ac);

%  sub-portfolio for the long leg
port_v3_long(tmp_long)=port1(tmp_long)
```

```
port_v3_long =
         0         0         0    0.0452         0    0.1142         0         0
0         0
```

```
port_v3_long=port_v3_long/sum(port_v3_long)
```

```
port_v3_long =
         0         0         0    0.2836         0    0.7164         0         0
0         0
```

```
%  sub-portfolio for the short leg
port_v3_short(tmp_short)=port1(tmp_short)
```

```
port_v3_short =
         0         0         0         0         0         0         0    0.0894
0.0167         0
```

```
port_v3_short=port_v3_short/sum(port_v3_short)
```

```
port_v3_short =
         0         0         0         0         0         0         0    0.8427
0.1573         0
```

```
%  view portfolio
port_v3=port_v3_long-port_v3_short
```

```
port_v3 =
         0         0         0    0.2836         0    0.7164         0   -0.8427   -
0.1573         0
```

```
%  the view portfolio's expected return
q3=0.30;

%  compared to equlibrium return of the view portfolio
fprintf('View 3: equlibrium return=%.2f%%, active view
return=%.2f%%.\n',port_v3*Pi,q3)
```

```
View 3: equlibrium return=0.03%, active view return=0.30%.
```

```
%  the view matrix is a kxN matrix where each row corresponds to a view
%  portfolio
P=[port_v1;port_v2;port_v3]
```

```
P =
         0    1.0000         0         0         0         0         0         0         0
0
         0         0    1.0000         0   -1.0000         0         0         0
0         0
         0         0         0    0.2836         0    0.7164         0   -0.8427   -
0.1573         0
```

```
%  show the three view portfolios in table format
array2table(P','RowNames',ac_name,'VariableNames',{'View1','View2','View3'})
ans = 10×3 table
```

	View1	View2	View3
DSCR	0	0	0
ENER	1	0	0
FINA	0	1	0
HLTH	0	0	0.28358
INDU	0	-1	0
INFT	0	0	0.71642
MATS	0	0	0
REAL	0	0	-0.84271
STPL	0	0	-0.15729
TCOM	0	0	0

```
%  View Return Vector Q
Q=[q1;q2;q3]
```

```
Q =
    0.2000
    0.0500
    0.3000
```

```
%  calculate the first diagonal element (corresponding to View 1)
omega1=tau*(port_v1*Sigma*port_v1');
%  ... (.. to View 2)
omega2=tau*(port_v2*Sigma*port_v2');
%  ... (.. to View 3)
omega3=tau*(port_v3*Sigma*port_v3');

%  the View uncertainty matrix
Omega=diag([omega1,omega2,omega3])
```

```
Omega =
   27.1971         0         0
         0   10.3325         0
         0         0   13.5179
```

```
%  calculate C
C=tau*Sigma;

%  calculate H
H=P'*inv(Omega)*P+inv(C);

%  calculate nu
nu=inv(H)*(P'*inv(Omega)*Q+inv(C)*Pi);
% calculate the new Sigma
Sigma = H^-1+Sigma
```

```
Sigma =
  396.1261  155.1036  258.1310  235.6896  276.2334  259.8539  268.2503  151.3274
190.5943  244.1126
  155.1036  489.1678  157.4639  137.3481  199.7745  193.2881  215.3107   61.4781
121.4877  191.1677
  258.1310  157.4639  389.9942  222.0443  265.8076  206.9760  256.1640  158.6203
189.4215  226.4604
  235.6896  137.3481  222.0443  275.2450  219.8426  217.9835  211.1580  143.8103
168.8351  209.8064
  276.2334  199.7745  265.8076  219.8426  327.5039  250.3967  283.4173  133.1300
176.7619  226.7777
  259.8539  193.2881  206.9760  217.9835  250.3967  322.3089  246.3729  121.1230
169.0449  242.5109
  268.2503  215.3107  256.1640  211.1580  283.4173  246.3729  352.2347  130.9309
177.9600  237.6213
  151.3274   61.4781  158.6203  143.8103  133.1300  121.1230  130.9309  288.2811
97.0268  121.9492
  190.5943  121.4877  189.4215  168.8351  176.7619  169.0449  177.9600   97.0268
186.6153  184.4702
  244.1126  191.1677  226.4604  209.8064  226.7777  242.5109  237.6213  121.9492
184.4702  327.7259
```

```
%   let's compare the equlibrium return Pi and the updated expected return
%   nu
array2table([Pi,nu],'RowNames',ac_name,'VariableNames',{'ORIGINAL_Exp_Ret','Upd
ated_Exp_Ret'})
ans = 10×2 table
            ORIGINAL_Exp_Ret    Updated_Exp_Ret
            ________________    _______________

    DSCR        0.55391             0.59471
    ENER        0.19217              0.2056
    FINA       -0.11032            0.082348
    HLTH        0.40121             0.46234
    INDU        0.28657             0.27458
    INFT        0.57578             0.63064
    MATS        0.43587             0.46592
    REAL        0.55038             0.49262
    STPL        0.17996             0.23509
    TCOM       0.080301             0.14794
```

```
%Observations: except the TWO sectors(INDU,REAL) underperform the original
return, all the other sectors outperform the original ones,
% which is the impact of active views for the improvement of portfolio
performance
%   the optimal weight vector from BL model
%optimal portfolio with active view
% annulized volatility of portfolio not greater than 10%

%   objective function
obj_func1=@(w) -w'*nu
obj_func1 = 包含以下值的 function_handle:
    @(w)-w'*nu
```

```
%  use the equal-weighted portfolio as starting point
w0=ones(num_ac,1)/num_ac;

%   we need to add a non-linear constraint, which is:
%   w'*Sigma*w<=(10/sqrt(12))^2
%   run the optimization
[w_opt2,fval]=fmincon(obj_func1,w0,A,b,Aeq,beq,[],[],@(w)
nonlcon(w,Sigma,(10/sqrt(12))^2))
```

```
   Local minimum found that satisfies the constraints.

   Optimization completed because the objective function is non-decreasing in
   feasible directions, to within the default value of the optimality tolerance,
   and constraints are satisfied to within the default value of the constraint
   tolerance.

   <stopping criteria details>

   w_opt2 =
       0.0688
       0.0020
      -0.0732
       0.0569
      -0.1268
       0.1438
       0.0869
```

```
    0.0665
    0.0124
   -0.1302


fval = -0.1744
```

```
w_opt=w_opt2;

%  check if net leverage of 1
fprintf('Net Leverage of The BL Optimal Portfolio = %.1f%%.\n',100*sum(w_opt));
```

```
Net Leverage of The BL Optimal Portfolio = 10.7%.
```

```
%  let's compare with the equlibrium portfolio, i.e., the market portfolio
%  which is the optimal one without active views
array2table([port1,w_opt]*100,'RowNames',ac_name,'VariableNames',{'PORT1','PORT
2'})
```

ans = 10×2 table

	PORT1	PORT2
DSCR	6.5342	6.8828
ENER	0.19393	0.19508
FINA	-12.577	-7.3153
HLTH	4.5207	5.6918
INDU	-6.404	-12.679
INFT	11.421	14.379
MATS	8.2519	8.6922
REAL	8.9361	6.6494
STPL	1.6679	1.2411
TCOM	-12.356	-13.015

NOTE:

1. PORT 1 is the original portfolio without active view, PORT 2 is the new portfolio with active view
2. Non-linear constraint function(nonlcon) for the portfolio volatility, is nonlcon=@circlecon, where circlecon is :

```
function [c,ceq] = circlecon(w,Sigma,Var_Budget)
c = w'*Sigma*w-Var_Budget;
ceq = [];
end
```