RELIABILITY IN PORTFOLIO OPTIMIZATION USING UNCERTAIN ESTIMATES

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Technology

by

RACHIT SETH

Y6114008

under the supervision of

Dr. RAGHU NANDAN SENGUPTA

Department of Industrial and Management Engineering

DEPARTMENT OF INDUSTRIAL AND MANAGEMENT ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

May, 2008

CERTIFICATE

This is to certify that the thesis titled **"Reliability in Portfolio Optimization using Uncertain Estimates"**, submitted by Rachit Seth to the Indian Institute of Technology Kanpur in partial fulfillment of the requirement for the degree of Master of Technology, is a record of bona-fide research work, carried out by him under my supervision. The results embodied in the thesis have not been submitted to any other university or institution for the award of any degree or diploma.

Dr. Raghu Nandan Sengupta

Thesis Supervisor

Assistant Professor

Department of Industrial & Management Engineering

Indian Institute of Technology Kanpur

Kanpur 208016

May 2008

Model I

Code 1: optimize_main.m % This is the main code which calls the other functions and codes.

```
% The code is to perform the optimization using SORA for Model 1.
% MeanER contains the mean of  expected returns from bootstrap
% SigmaER contains the standard deviation of expected returns from bootstrap
% CovER contains the covariance matrix for expected returns from bootstrap
% MeanN contains the mean of the weights.
% SigmaN contains the standard deviation of weights.
% CovN contains the covariance matrix for weights.
% MeanR contains the mean of returns
% SigmaR contains the standard deviation of returns
% CovR contains the covariance matrix for returns
% CorrC is the correlation matrix from the Cov matrix of returns
% R_mpp1 : Mpp corresponding to MeanR (parameter Mpp acc to sora)
% ub and lb are vector containing upper and lower bound
% Stk contains the last traded price of the scripts
% % confun.m contains the nonlinear constraints
% objective.m contains the objective function

clc;clear all;
global MeanR MeanN MeanER SigmaR SigmaN SigmaER CovR CovN CovER CorrC stk R_mpp1 optimvalue q
%------------------------------------------------------------------------%
%             Input parameters to read from Xls file                     %
%------------------------------------------------------------------------%
filename = 'data.xls'; %the xls filename
%---------------------------  for returns ------------------------------%
sheet = 'InputReturns';      % The name of the sheet in the file above
[CovR, header] = xlsread(filename, sheet, 'A1:J10');
[MeanR] = xlsread(filename, sheet, 'B14:J14');
for i=1:9
   for j=1:i
      CovR(i,j)=CovR(j,i);
   end
end
```

```
[SigmaR, CorrC] = cov2corr(CovR);
[stk] = xlsread(filename, sheet, 'B21:J21');
%---------------------------  for weights ------------------------------ %
sheet = 'InputWeights';      % The name of the sheet in the file above
[CovN, header] = xlsread(filename, sheet, 'A1:J10');
[MeanN] = xlsread(filename, sheet, 'B14:J14');
for i=1:9
   for j=1:i
      CovN(i,j)=CovN(j,i);
   end
end
[SigmaN, CorrN] = cov2corr(CovN);
%---------------------- for expected returns from bootstrap -------------------------------%
filename = 'bootstrapdata_nine_scrips.xls'; %the xls filename
sheet = 'InpExpreturns';      % The name of the sheet in the file above
[dataER, header] = xlsread(filename, sheet, 'A2:I502');
CovER = cov(dataER);
MeanER = mean(dataER);
[SigmaER, CorrER] = cov2corr(CovER);
%---------------------- for standard deviations from bootstrap -------------------------------%
filename = 'bootstrapdata_nine_scrips.xls'; %the xls filename
sheet = 'InpSDreturns';      % The name of the sheet in the file above
[dataSD, header] = xlsread(filename, sheet, 'K2:S502');
CovSD = cov(dataSD);
MeanSD = mean(dataSD);
[SigmaSD, CorrSD] = cov2corr(CovSD);
%------------------ Starting the loop to vary initial amount----------
%v0=0;
%for g = 1:100
  %  v0=v0+10000;
%-----------------  END of Inputs  from XLs sheet ------------------------%
R_mpp1 = MeanR; % For first cycle the MPP value is the Mean Value
%----------------------NEWBNB-------------------------------------------%
fun        = 'objective_case1';
x0         = [1500 1500 1500 1500 1500 1500 1500 1500 1500]'; % Starting Point
xstat      = [1 1 1 1 1 1 1 1 1]';% With reference to BNB code indicating that all variables are integers
xl         = [0 0 0 0 0 0 0 0 0]';
xu         = [5000 5000 5000 5000 5000 5000 5000 5000 5000]';%Upper bound
```

```
v0=1000000;% Initial Endowment
% Now Incorporating the 30% of initial endowment constraint
%-------------------------------------------------------------
v1=.3*v0;
b         = [v0 0 v1 v1 v1 v1 v1 v1 v1 v1]';
a(1,1:9)   = stk;
a(2,1:9)      =  [(-1*stk(1)*(1+R_mpp1(1)))  (-1*stk(2)*(1+R_mpp1(2)))  (-1*stk(3)*(1+R_mpp1(3)))  (-1*stk(4)*(1+R_mpp1(4)))  (-1*stk(5)*(1+R_mpp1(5)))  (-
1*stk(6)*(1+R_mpp1(6))) (-1*stk(7)*(1+R_mpp1(7))) (-1*stk(8)*(1+R_mpp1(8))) (-1*stk(9)*(1+R_mpp1(9)))]  ;

a(3,1:9)   = [76.8 0 0 0 0 0 0 0 0];
a(4,1:9)   = [ 0 45.07 0 0 0 0 0 0 0];
a(5,1:9)   = [0 0 107.2 0 0 0 0 0 0];
a(6,1:9)   = [0 0 0 110.7 0 0 0 0 0];
a(7,1:9)   = [0 0 0 0 110.08 0 0 0 0];
a(8,1:9)   = [0 0 0 0 0 21.78 0 0 0];
a(9,1:9)   = [0 0 0 0 0 0 35.19 0 0];
a(10,1:9)   = [0 0 0 0 0 0 0 13.68 0];
a(11,1:9)   = [0 0 0 0 0 0 0 0 84.72];
%-------------------------------------------------------------
aeq       = [];
beq       = [];
nonlc      = 'confun_case1';
setts     = [];
opts      = optimset('display','off','MaxSQPIter',1000);
iter = 0;
fval=0;
[errmsg,Z(iter+1),X(:,iter+1),t,c,fail]=bnb_ml60(fun,x0,xstat,xl,xu,a,b,aeq,beq,nonlc,setts,opts);% Calling BNB code for optimization
disp('solution:'), X
y(iter+1,:)=X(:,iter+1).'; % getting row vector corresponding to column vector x obtained above
r1=0;
r2=0;
ri1=0;
ri2=0;
for i = 1 :9
   r2 = r2+ y(iter+1,i)*stk(i)*(1+ MeanR(i));%ExpectedValueofPortfolio
   r1 = r1 + y(iter+1,i)*stk(i)*(1+R_mpp1(i)); %ExpectedValueofPortfolio
end
for i = 1 :9
```

```
    for j = 1 :9
        ri2 = ri2 + y(iter+1,i)*y(iter+1,j)*stk(i)*stk(j)*CorrC(i,j)*SigmaR(i)*SigmaR(j);
    end
end

ExpValueofPortfolio(1,iter+1)= r1;
ExpValueofPortfolio_MeanR(1,iter+1)= r2;
Risk_SigmaR(1,iter+1)=sqrt(ri2);

while((abs(Z(iter+1)-fval)>0.01))
    optimvalue = y(iter+1,:);   % for new mpp approach
    fval=Z(iter+1);
    x0 = X(:,iter+1);%Changing the starting point to the optimal solution obtained in the previous cycle
    iter = iter+1;
    % With the solution found above find the corresponding MPP point
    % using the row vector y, as x was in form of column vector
    f2_mpp(iter,:) = getmpp_pma(R_mpp1,y(iter,:));  % Calling the MPP function
    R_mpp1=f2_mpp(iter,:);
    % perform optimization again
    % Calling BNB code for optimization
    [errmsg,Z(iter+1),X(:,iter+1),t,c,fail]=bnb_ml60(fun,x0,xstat,xl,xu,a,b,aeq,beq,nonlc,setts,opts);
    disp('solution:'), X
    y(iter+1,:)=X(:,iter+1).'; % getting row vector corresponding to column vector x obtained above
    r1=0;
    r2=0;
    ri1=0;
    ri2=0;
    for i = 1 :9
      r2 = r2+ y(iter+1,i)*stk(i)*(1+ MeanR(i));%ExpectedValueofPortfolio
      r1 = r1 + y(iter+1,i)*stk(i)*(1+R_mpp1(i)); %ExpectedValueofPortfolio
    end
   for i = 1 :9
     for j = 1 :9
        ri2 = ri2 + y(iter+1,i)*y(iter+1,j)*stk(i)*stk(j)*CorrC(i,j)*SigmaR(i)*SigmaR(j);
     end
   end

ExpValueofPortfolio(1,iter+1)= r1;
```

```
ExpValueofPortfolio_MeanR(1,iter+1)= r2;
Risk_SigmaR(1,iter+1)=sqrt(ri2);
end
%-------------------------------------------------------------------------%
%                       OPTIMIZATION ENDS                                 %
%-------------------------------------------------------------------------%
```

Code 2: Objective Function objective.m being called by optimize_main.m

```
% the objective function('objective.m')
function f = objective_case1(x)
global MeanR MeanN MeanER MeanRmpp MeanRC SigmaR SigmaN SigmaER CovR CovN CovER CorrC SM1 SM2 stk R_mpp1
a=0;
% Return
for i = 1:9
    a = a + x(i)*MeanER(i);
end
f = -1*a;  % Maximize (returns)
```

Code 3: Constraint Function confun.m being called by optimize_main.m

```
% contraint functions('confun.m')
function [cin,ceq] = confun_case1(x)
global MeanR MeanN MeanER MeanRmpp MeanRC SigmaR SigmaN SigmaER CovR CovN CovER CorrC SM1 SM2 stk R_mpp1
f0 = 0;
f1 = 0;
f12 =0;
fp1 = 0;
z= -2.88; % Corresponding to alpha_VaR
r = 0;
for i = 1 :9
    f0 = f0 + x(i)*stk(i);% Total Investment
    f1 = f1+ x(i)*stk(i)*(1+ MeanR(i));
end
```

Code 4: MPP calculation Code getmpp_pma.m being called by optimize_main.m

```
% The code is to get the mpp point corresponding to first prob constraint
function r_mpp = getmpp_pma(mpppt,x)
global MeanR MeanN MeanER MeanRC SigmaR SigmaN SigmaER CovR CovN CovER CorrC stk R_mpp1 optimvalue
%------------------ Rosenblatt Transformation ----------------------------%
u1 = RTrans_case1(mpppt(1:9),MeanER,CovER); % Function being called
mpppt(1:9)
u1
%--------------------------   End   ------------------------------------%
%----------------------- MPP Calculation ---------------------------------%
settings=[];
options = optimset('MaxFunEvals',10000,'MaxSQPIter',1000);
a = u1;
xlb=[.00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001];
xub=[.99 .99 .99 .99 .99 .99 .99 .99 .99];
i=2;
x0 = a;
% Optimization for PMA
[u1(i,:),Z(i),exitflag] = fmincon('objective_mpp',x0,[],[],[],[],xlb,xub,'confun_mpp',options);
%--------------------------   End   ------------------------------------%
%------------------ Inverse Rosenblatt Transformation ---------------------%
r_mpp = RTransInv_case1(u1(i,1:9),MeanER,CovER);
u1(i,1:9)
r_mpp
%--------------------------   End   ------------------------------------%
```

Code 5: Objective Function objective_mpp.m being called by getmpp_pma.m

```
% Objective function for MPP calculation
function p = objective_mpp(u1)
global MeanR MeanN MeanER MeanRmpp MeanRC SigmaR SigmaN SigmaER CovR CovN CovER CorrC SM1 SM2 stk R_mpp1 optimvalue q
p=0;
q=0;
for i = 1:9
      q = q + optimvalue(i)*stk(i)*(1+u1(i)); % first probabilistic constraint
```

```
end
p=q;
disp(q);
```

Code 6: Constraint Function confun_mpp.m being called by getmpp_pma.m

---

```
function [cin,ceq] = confun_mpp(u1)
global MeanR MeanN MeanER MeanRmpp MeanRC SigmaR SigmaN SigmaER CovR CovN CovER CorrC R_mpp1 optimvalue
beta = 1.95;
sum = 0;
   for i = 1:9
      sum = sum + u1(i)*u1(i);
   end
   c1 = sqrt(sum);
ceq = [c1-beta];
cin = [];
```

Code 7: RTrans_case1.m called by getmpp_pma.m

```
%The following Code calcultes the Rosenbat Transformation
function U = RTrans_case1(x,Mean,Cov)
%---------------------- Input values ----------------------------------%
r = 9; %no of scrips
Result_Vector =[];
initial_weight = x;
%-------------------------   End   ----------------------------------%
%-------------------  Start of Transformation  -------------------------%
Result_Vector(1)= (initial_weight(1)-Mean(1))/sqrt(Cov(1,1));
  for k = 2:r
      matrix(1:k,1:k)= Cov(1:k,1:k);
      numer1 =0;
      for j= 1:(k-1)
         numer1 = numer1 + ((cofactor(matrix,k,j)/cofactor(matrix,k,k))*(initial_weight(j)-Mean(j)));
      end
      numer = (initial_weight(k)-Mean(k))+ numer1;
      denom = sqrt(det(matrix)/cofactor(matrix,k,k));
```

```
      Result_Vector(k)= (numer/denom);
   end

   for i=1:9
     U(i) = cdf('Normal',Result_Vector(i),Mean(i),sqrt(Cov(i,i)));
   end
%---------------------  End of Transformation  ---------------------------%
```

Code 8: RTransInv_case1.m called by getmpp_pma.m

```
-------------------------------------------------------------------------
%The following Code calcultes Inverse Rosenbat Transformation
function X = RTransInv_case1(u,Mean,Cov)
%------------------------ Input values ----------------------------------%
r = 9; %no of scrips
%--------------------------   End   ----------------------------------%
%-----------------  Start of  Inverse Transformation  ------------------%
X(1) = norminv(u(1),0,1)*sqrt(Cov(1,1))+Mean(1);
   for k = 2:r
      matrix(1:k,1:k)= Cov(1:k,1:k);
      numer1 =0;
      for j= 1:(k-1)
         numer1 = numer1 + ((cofactor(matrix,k,j)/cofactor(matrix,k,k))*(u(j)-Mean(j)));
      end
      denom = sqrt(det(matrix)/cofactor(matrix,k,k));
      X(k)= norminv(u(k),0,1)*denom - numer1 + Mean(k);
   end
%------------------  End of  Inverse Transformation  ---------------------%
```

Model II

Code 1: optimize_main.m % This is the main code which calls the other functions and codes.

```
% The code is to perform the optimization using SORA for Model 2.
% MeanER contains the mean of  expected returns from bootstrap
% SigmaER contains the standard deviation of expected returns from bootstrap
% CovER contains the covariance matrix for expected returns from bootstrap
% MeanN contains the mean of the weights.
% SigmaN contains the standard deviation of weights.
% CovN contains the covariance matrix for weights.
% MeanR contains the mean of returns
% SigmaR contains the standard deviation of returns
% CovR contains the covariance matrix for returns
% CorrC is the correlation matrix from the Cov matrix of returns
% % ub and lb are vector containing upper and lower bound
% Stk contains the last traded price of the scripts
% % confun.m contains the nonlinear constraints
% objective.m contains the objective function
% R_mpp1 and R_mpp2 contain the mpp values corresponding to first and second probabilistic constraints
clc;clear all;
global MeanR MeanN MeanER MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC R_mpp1 R_mpp2 optimvalue q
%-------------------------------------------------------------------------%
%             Input parameters to read from Xls file                      %
%-------------------------------------------------------------------------%
filename = 'data.xls'; %the xls filename
%----------------------------  for returns -------------------------------%
sheet = 'InputReturns';      % The name of the sheet in the file above
[CovR, header] = xlsread(filename, sheet, 'A1:J10');
[MeanR] = xlsread(filename, sheet, 'B14:J14');
for i=1:9
   for j=1:i
      CovR(i,j)=CovR(j,i);
   end
end
[SigmaR, CorrC] = cov2corr(CovR);
```

```
[stk] = xlsread(filename, sheet, 'B21:J21');
%---------------------------  for weights ----------------------------- %
sheet = 'InputWeights';      % The name of the sheet in the file above
[CovN, header] = xlsread(filename, sheet, 'A1:J10');
[MeanN] = xlsread(filename, sheet, 'B14:J14');
for i=1:9
   for j=1:i
      CovN(i,j)=CovN(j,i);
   end
end
[SigmaN, CorrN] = cov2corr(CovN);
%----------------------- for expected returns from bootstrap ----------%
filename = 'bootstrapdata_nine_scrips.xls'; %the xls filename
sheet = 'InpExpreturns';      % The name of the sheet in the file above
[dataER, header] = xlsread(filename, sheet, 'A2:I502');
CovER = cov(dataER);
MeanER = mean(dataER);
[SigmaER, CorrER] = cov2corr(CovER);
%----------------------- for standard deviations from bootstrap -----------%
filename = 'bootstrapdata_nine_scrips.xls'; %the xls filename
sheet = 'InpSDreturns';      % The name of the sheet in the file above
[dataSD, header] = xlsread(filename, sheet, 'K2:S502');
CovSD = cov(dataSD);
MeanSD = mean(dataSD);
[SigmaSD, CorrSD] = cov2corr(CovSD);
%------------------ Starting the loop to vary initial amount----------
%v0=0;
%for g = 1:100
  %  v0=v0+10000;
%-----------------  END of Inputs  from XLs sheet ------------------------%
% Putting Intitial MPP values as the means
R_mpp1 = MeanR;
R_mpp2 = MeanR;
%----------------------NEWBNB-----------------------------------------%
%-------------------------------------------------------------------------
fun        = 'objective_case1';
x0         = [1500 1500 1500 1500 1500 1500 1500 1500 1500]';
xstat      = [1 1 1 1 1 1 1 1 1]'; % Input to BNB Code
```

```
xl        = [0 0 0 0 0 0 0 0 0 0]';
xu        = [5000 5000 5000 5000 5000 5000 5000 5000 5000 5000]';
v0=1000000;
v1=.3*v0;
b         = [v0 0 v1 v1 v1 v1 v1 v1 v1 v1]';
a(1,1:9)   = stk;
a(2,1:9)      =  [(-1*stk(1)*(1+R_mpp1(1)))  (-1*stk(2)*(1+R_mpp1(2)))  (-1*stk(3)*(1+R_mpp1(3)))  (-1*stk(4)*(1+R_mpp1(4)))  (-1*stk(5)*(1+R_mpp1(5)))  (-
1*stk(6)*(1+R_mpp1(6))) (-1*stk(7)*(1+R_mpp1(7))) (-1*stk(8)*(1+R_mpp1(8))) (-1*stk(9)*(1+R_mpp1(9)))]  ;
a(3,1:9)   = [76.8 0 0 0 0 0 0 0 0];
a(4,1:9)   = [ 0 45.07 0 0 0 0 0 0 0];
a(5,1:9)   = [0 0 107.2 0 0 0 0 0 0];
a(6,1:9)   = [0 0 0 110.7 0 0 0 0 0];
a(7,1:9)   = [0 0 0 0 110.08 0 0 0 0];
a(8,1:9)   = [0 0 0 0 0 21.78 0 0 0];
a(9,1:9)   = [0 0 0 0 0 0 35.19 0 0];
a(10,1:9)   = [0 0 0 0 0 0 0 13.68 0];
a(11,1:9)   = [0 0 0 0 0 0 0 0 84.72];
aeq       = [];
beq       = [];
nonlc      = 'confun_case1';
setts     = [];
opts      = optimset('display','off','MaxSQPIter',1000);
%------------------------------------------------------------------------%
%                 OPTIMIZATION PART STARTS HERE                          %
%------------------------------------------------------------------------%
iter = 0;
fval=0;
% Now calling BNB code for optimization
[errmsg,Z(iter+1),X(:,iter+1),t,c,fail]=bnb_ml60(fun,x0,xstat,xl,xu,a,b,aeq,beq,nonlc,setts,opts);
disp('solution:'), X
y(iter+1,:)=X(:,iter+1).'; % getting row vector corresponding to column vector x obtained above
r1=0;
r2=0;
ri1=0;
ri2=0;
for i = 1 :9
   r2 = r2+ y(iter+1,i)*stk(i)*(1+ MeanR(i));%ExpectedValueofPortfolio
   r1 = r1 + y(iter+1,i)*stk(i)*(1+R_mpp1(i)); %ExpectedValueofPortfolio
```

```
end
for i = 1 :9
   for j = 1 :9
      %ri1 = ri1 + y(iter+1,i)*y(iter+1,j)*stk(i)*stk(j)*CorrC(i,j)*SD_mpp1(i+9)*SD_mpp1(j+9);
      ri2 = ri2 + y(iter+1,i)*y(iter+1,j)*stk(i)*stk(j)*CorrC(i,j)*SigmaR(i)*SigmaR(j);
   end
end
ExpValueofPortfolio(1,iter+1)= r1;
ExpValueofPortfolio_MeanR(1,iter+1)= r2;
Risk_SigmaR(1,iter+1)=sqrt(ri2);
while((abs(Z(iter+1)-fval)>0.01))
   optimvalue = y(iter+1,:);   % for new mpp approach
   fval=Z(iter+1);
   x0 = X(:,iter+1);%Changing the starting point to the optimal solution obtained in the previous cycle
   iter = iter+1;
   % With the solution found above find the corresponding MPP point
   % using the row vector y, as x was in form of column vector
   % Calling the MPP functions
   f2_mpp(iter,:) = getmpp_pma(R_mpp1,y(iter,:));
   R_mpp1=f2_mpp(iter,:);
   f3_mpp(iter,:)= getmppR2_pma(R_mpp2,y(iter,:));
   R_mpp2=f3_mpp(iter,:);
   % Perform optimization again , calling BNB code for optimization
   [errmsg,Z(iter+1),X(:,iter+1),t,c,fail]=bnb_ml60(fun,x0,xstat,xl,xu,a,b,aeq,beq,nonlc,setts,opts);
   disp('solution:'), X
   y(iter+1,:)=X(:,iter+1).'; % getting row vector corresponding to column vector x obtained above
   r1=0;
   r2=0;
   ri1=0;
   ri2=0;
   for i = 1 :9
     r2 = r2+ y(iter+1,i)*stk(i)*(1+ MeanR(i));%ExpectedValueofPortfolio
     r1 = r1 + y(iter+1,i)*stk(i)*(1+R_mpp1(i)); %ExpectedValueofPortfolio
   end
   for i = 1 :9
      for j = 1 :9

        ri2 = ri2 + y(iter+1,i)*y(iter+1,j)*stk(i)*stk(j)*CorrC(i,j)*SigmaR(i)*SigmaR(j);
```

```
      end
   end

   ExpValueofPortfolio(1,iter+1)= r1;
   ExpValueofPortfolio_MeanR(1,iter+1)= r2;
   Risk_SigmaR(1,iter+1)=sqrt(ri2);

end
%---------------------------------------------------------------------------%
%                        OPTIMIZATION ENDS                                  %
%---------------------------------------------------------------------------%
```

Code 2: Objective Function objective.m being called by optimize_main.m

```
% the objective function('objective.m')
function f = objective_case1(x)
global MeanR MeanN MeanER MeanRmpp MeanRC SigmaR SigmaN SigmaER CovR CovN CovER CorrC SM1 SM2 stk R_mpp1 R_mpp2
a=0;
% Return
for i = 1:9
   a = a + x(i)*MeanER(i);
end
f = -1*a;  % Maximize (returns)
```

Code 3: Constraint Function confun.m being called by optimize_main.m

```
% contraint functions('confun.m')
function [cin,ceq] = confun_case1(x)
global MeanR MeanN MeanER MeanRmpp MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC R_mpp1   R_mpp2
SD_mpp1
f0 = 0;
f1 = 0;
f12 =0;
fp1 = 0;
z= -2.88;
r = 0;
```

```
for i = 1 :9
   f0 = f0 + x(i)*stk(i);% Total Investment
   f1 = f1+ x(i)*stk(i)*(1+ MeanR(i));
   fp1 = fp1 + x(i)*stk(i)*(1+R_mpp2(i)); % second probabilistic constraint first part
end
% summation of variance involved in the second probabilistic constraint
for i = 1 :9
   for j = 1 :9
      f12 = f12 + x(i)*x(j)*stk(i)*stk(j)*CorrC(i,j)*SigmaR(i)*SigmaR(j);
   end
end
ceq = [];
cin = [-fp1+z*sqrt(f12)+f0*(1-0.05)];
f0
```

Code 4: MPP calculation Code getmpp_pma.m being called by optimize_main.m

---

```
% The code is to get the mpp point corresponding to first prob constraint
function r_mpp = getmpp_pma(mpppt,x)
global MeanR MeanN MeanER MeanRmpp MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC R_mpp1 R_mpp2
optimvalue
%------------------ Rosenblatt Transformation ----------------------------%
u1 = RTrans_case1(mpppt(1:9),MeanER,CovER);
mpppt(1:9)
u1
%--------------------------   End   ------------------------------------%
%----------------------- MPP Calculation ---------------------------------%
settings=[];
options = optimset('MaxFunEvals',10000,'MaxSQPIter',1000);
a = u1;
xlb=[.00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001];
xub=[.99 .99 .99 .99 .99 .99 .99 .99 .99];
i=2;
x0 = a;

% Performing Optimization for PMA
```

```
[u1(i,:),Z(i),exitflag] = fmincon('objective_mpp',x0,[],[],[],[],xlb,xub,'confun_mpp',options);
%-------------------------   End   -----------------------------------%
%---------------- Inverse Rosenblatt Transformation ---------------------%
r_mpp = RTransInv_case1(u1(i,1:9),MeanER,CovER);
u1(i,1:9)
r_mpp
%-------------------------   End   -----------------------------------%
```

Code 5: Objective Function objective_mpp.m being called by getmpp_pma.m

---

```
Mpp1 Objective function
function p = objective_mpp(u1)
global MeanR MeanN MeanER MeanRmpp MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC R_mpp1 R_mpp2
optimvalue q
p=0;
q=0;
for i = 1:9
      q = q + optimvalue(i)*stk(i)*(1+u1(i)); % first probabilistic constraint
end
p=q;
disp(q);
```

Code 6: Constraint Function confun_mpp.m being called by getmpp_pma.m

---

```
% Mpp called constraint function
function [cin,ceq] = confun_mpp(u1)
global MeanR MeanN MeanER MeanRmpp MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC R_mpp1  R_mpp2
SD_mpp1 optimvalue
beta = 2.33;
sum = 0;
   for i = 1:9
      sum = sum + u1(i)*u1(i);
   end
   c1 = sqrt(sum);
ceq = [c1-beta];cin = [];
```

Code 7: RTrans_case1.m called by getmpp_pma.m

```
%The following Code calcultes the Rosenbat Transformation
function U = RTrans_case1(x,Mean,Cov)
%------------------------ Input values ----------------------------------%
r = 9; %no of scrips
Result_Vector =[];
initial_weight = x;
%-------------------------   End   ----------------------------------%
%--------------------  Start of Transformation  -------------------------%
Result_Vector(1)= (initial_weight(1)-Mean(1))/sqrt(Cov(1,1));
  for k = 2:r
      matrix(1:k,1:k)= Cov(1:k,1:k);
      numer1 =0;
      for j= 1:(k-1)
         numer1 = numer1 + ((cofactor(matrix,k,j)/cofactor(matrix,k,k))*(initial_weight(j)-Mean(j)));
      end
      numer = (initial_weight(k)-Mean(k))+ numer1;
      denom = sqrt(det(matrix)/cofactor(matrix,k,k));
      Result_Vector(k)= (numer/denom);
   end

   for i=1:9
     U(i) = cdf('Normal',Result_Vector(i),Mean(i),sqrt(Cov(i,i)));
   end
%---------------------  End of Transformation  --------------------------%
```

Code 8: RTransInv_case1.m called by getmpp_pma.m

```
%The following Code calcultes Inverse Rosenbat Transformation
function X = RTransInv_case1(u,Mean,Cov)
%------------------------ Input values ----------------------------------%
r = 9; %no of scrips
%-------------------------   End   ----------------------------------%
%-----------------  Start of  Inverse Transformation  ------------------%
```

```
X(1) = norminv(u(1),0,1)*sqrt(Cov(1,1))+Mean(1);
   for k = 2:r
      matrix(1:k,1:k)= Cov(1:k,1:k);
      numer1 =0;
      for j= 1:(k-1)
         numer1 = numer1 + ((cofactor(matrix,k,j)/cofactor(matrix,k,k))*(u(j)-Mean(j)));
      end
      denom = sqrt(det(matrix)/cofactor(matrix,k,k));
      X(k)= norminv(u(k),0,1)*denom - numer1 + Mean(k);
   end
%------------------  End of  Inverse Transformation  ----------------------%
```

Code 9: MPP calculation Code getmppR2_pma.m being called by optimize_main.m

```
% The code is to get the mpp point corresponding to second prob constraint
function R2_mpp = getmppR2_pma(mpppt1,x1)
global MeanR MeanN MeanER MeanRmpp MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC R_mpp1 R_mpp2
optimvalue
%------------------ Rosenblatt Transformation ----------------------------%
s1(1:9)  = RTrans_case1(mpppt1(1:9),MeanER,CovER);
mpppt1(1:9)
s1
%--------------------------   End   ------------------------------------%
%---------------------- MPP Calculation ---------------------------------%
settings=[];
options = optimset('MaxFunEvals',10000,'MaxSQPIter',1000);
a = s1;
xlb=[.00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001];
xub=[.99 .99 .99 .99 .99 .99 .99 .99 .99];
i=2;
x0 = a;
% Performimg Optimization for PMA
[s1(i,:),Z1(i),exitflag] = fmincon('objective_mpp_R2',x0,[],[],[],[],xlb,xub,'confun_mpp_R2',options);
%--------------------------   End   ------------------------------------%
%----------------- Inverse Rosenblatt Transformation ---------------------%
R2_mpp(1:9) = RTransInv_case1(s1(i,1:9),MeanER,CovER);
s1(i,1:9)
```

```
R2_mpp
%-------------------------    End    ----------------------------------%
```

Code 10: Objective Function objective_mpp_R2.m being called by getmppR2_pma.m

---

```
% Mpp2 Objective function
function p = objective_mpp_R2(s1)
global MeanR MeanN MeanER MeanRmpp MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrC CorrSD R_mpp1 R_mpp2
optimvalue q
p=0;
q0=0;
q1=0;
q2=0;
for i = 1:9
      q0 = q0 + optimvalue(i)*stk(i)*(1-0.05); %last termend
for i = 1:9
      q1 = q1 + optimvalue(i)*stk(i)*(1+s1(i)); %first term which contains the uncertain parameter
end
for i = 1:9
   for j = 1:9
      q2 = q2 + optimvalue(i)*optimvalue(j)*stk(i)*stk(j)*CorrSD(i,j)*SigmaR(i)*SigmaR(j);%second term
   end
end
q = q1+2.88*sqrt(q2)-q0;
p=q;
disp(q);
```

Code 11: Constraint Function confun_mpp_R2.m being called by getmppR2_pma.m

---

```
% Mpp2 called constraint function
function [cin,ceq] = confun_mpp_R2(s1)
global MeanR MeanN MeanER MeanRmpp MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC R_mpp1 R_mpp2
optimvalue
beta = 2.33;
```

```
c2   = 0;
sum  = 0;
   for i = 1:9
      sum = sum + s1(i)*s1(i);
   end
c2   = sqrt(sum);
ceq  = [c2-beta];
cin  = [];
```

Model III

Code 1 : optimize_main.m

```
% The code is to perform the optimization using SORA for Model 3.
% MeanER contains the mean of  expected returns from bootstrap
% SigmaER contains the standard deviation of expected returns from bootstrap
% CovER contains the covariance matrix for expected returns from bootstrap
% MeanN contains the mean of the weights.
% SigmaN contains the standard deviation of weights.
% CovN contains the covariance matrix for weights.
% MeanR contains the mean of returns
% SigmaR contains the standard deviation of returns
% CovR contains the covariance matrix for returns
% CorrC is the correlation matrix from the Cov matrix of returns
% % ub and lb are vector containing upper and lower bound
% Stk contains the last traded price of the scripts
% confun.m contains the nonlinear constraints
% objective.m contains the objective function
%R_mpp1 and SD_mpp1 contain the mpp values corresponding to first and second probabilistic constraints
clc;clear all;
global MeanR MeanN MeanER MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC  stk R_mpp1 SD_mpp1 optimvalue q

%-------------------------------------------------------------------------%
%            Input parameters to read from Xls file                       %
%-------------------------------------------------------------------------%
```

```
filename = 'data.xls'; %the xls filename
%--------------------------  for returns ------------------------------%
sheet = 'InputReturns';      % The name of the sheet in the file above
[CovR, header] = xlsread(filename, sheet, 'A1:J10');
[MeanR] = xlsread(filename, sheet, 'B14:J14');
for i=1:9
   for j=1:i
      CovR(i,j)=CovR(j,i);
   end
end
[SigmaR, CorrC] = cov2corr(CovR);
[stk] = xlsread(filename, sheet, 'B21:J21');
%--------------------------  for weights ------------------------------ %
sheet = 'InputWeights';      % The name of the sheet in the file above
[CovN, header] = xlsread(filename, sheet, 'A1:J10');
[MeanN] = xlsread(filename, sheet, 'B14:J14');
for i=1:9
   for j=1:i
      CovN(i,j)=CovN(j,i);
   end
end
[SigmaN, CorrN] = cov2corr(CovN);
%---------------------- for expected returns from bootstrap ------------------------------%
filename = 'bootstrapdata_nine_scrips.xls'; %the xls filename
sheet = 'InpExpreturns';      % The name of the sheet in the file above
[dataER, header] = xlsread(filename, sheet, 'A2:I502');
CovER = cov(dataER);
MeanER = mean(dataER);
[SigmaER, CorrER] = cov2corr(CovER);
%---------------------- for standard deviations from bootstrap ------------------------------%
filename = 'bootstrapdata_nine_scrips.xls'; %the xls filename
sheet = 'InpSDreturns';      % The name of the sheet in the file above
[dataSD, header] = xlsread(filename, sheet, 'K2:S502');
CovSD = cov(dataSD);
MeanSD = mean(dataSD);
[SigmaSD, CorrSD] = cov2corr(CovSD);
%------------------ Starting the loop to vary initial amount----------
%v0=0;
```

```
%for g = 1:100
 %   v0=v0+10000;
%----------------  END of Inputs  from XLs sheet ------------------------%
R_mpp1 = MeanR;
SD_mpp1 = cat(2,MeanR,MeanSD);
%----------------------NEWBNB-----------------------------------------%
%frontcon(MeanR,CovR, 5);
%-------------------------------------------------------------------
fun       = 'objective_case1';
x0        = [1500 1500 1500 1500 1500 1500 1500 1500 1500]';
xstat     = [1 1 1 1 1 1 1 1 1]';% Wrt to BNB
xl        = [0 0 0 0 0 0 0 0 0]';
xu        = [5000 5000 5000 5000 5000 5000 5000 5000 5000]';
v0=1000000;
v1=.3*v0;
b         = [v0 0 v1 v1 v1 v1 v1 v1 v1 v1]';
a(1,1:9)   = stk;
a(2,1:9)     =  [(-1*stk(1)*(1+R_mpp1(1)))  (-1*stk(2)*(1+R_mpp1(2)))  (-1*stk(3)*(1+R_mpp1(3)))  (-1*stk(4)*(1+R_mpp1(4)))  (-1*stk(5)*(1+R_mpp1(5)))  (-
1*stk(6)*(1+R_mpp1(6))) (-1*stk(7)*(1+R_mpp1(7))) (-1*stk(8)*(1+R_mpp1(8))) (-1*stk(9)*(1+R_mpp1(9)))]  ;
a(3,1:9)   = [76.8 0 0 0 0 0 0 0 0];
a(4,1:9)   = [ 0 45.07 0 0 0 0 0 0 0];
a(5,1:9)   = [0 0 107.2 0 0 0 0 0 0];
a(6,1:9)   = [0 0 0 110.7 0 0 0 0 0];
a(7,1:9)   = [0 0 0 0 110.08 0 0 0 0];
a(8,1:9)   = [0 0 0 0 0 21.78 0 0 0];
a(9,1:9)   = [0 0 0 0 0 0 35.19 0 0];
a(10,1:9)   = [0 0 0 0 0 0 0 13.68 0];
a(11,1:9)   = [0 0 0 0 0 0 0 0 84.72];
aeq       = [];
beq       = [];
nonlc     = 'confun_case1';
setts     = [];
opts      = optimset('display','off','MaxSQPIter',1000);

%-------------------------------------------------------------------------%
%                  OPTIMIZATION PART STARTS HERE                          %
%-------------------------------------------------------------------------%
iter = 0;
```

```
fval=0;
% Optimization by calling the BNB code
[errmsg,Z(iter+1),X(:,iter+1),t,c,fail]=bnb_ml60(fun,x0,xstat,xl,xu,a,b,aeq,beq,nonlc,setts,opts);
disp('solution:'), X
y(iter+1,:)=X(:,iter+1).'; % getting row vector corresponding to column vector x obtained above
r1=0;
r2=0;
ri1=0;
ri2=0;
for i = 1 :9
   r2 = r2+ y(iter+1,i)*stk(i)*(1+ MeanR(i));%ExpectedValueofPortfolio
   r1 = r1 + y(iter+1,i)*stk(i)*(1+SD_mpp1(i)); %ExpectedValueofPortfolio
end
for i = 1 :9
   for j = 1 :9
      ri1 = ri1 + y(iter+1,i)*y(iter+1,j)*stk(i)*stk(j)*CorrC(i,j)*SD_mpp1(i+9)*SD_mpp1(j+9);
      ri2 = ri2 + y(iter+1,i)*y(iter+1,j)*stk(i)*stk(j)*CorrC(i,j)*SigmaR(i)*SigmaR(j);
   end
end
ExpValueofPortfolio(1,iter+1)= r1;
RiskofPortfolio(1,iter+1)= sqrt(ri1);
ExpValueofPortfolio_MeanR(1,iter+1)= r2;
Risk_SigmaR(1,iter+1)=sqrt(ri2);
while((abs(Z(iter+1)-fval)>0.01))
   optimvalue = y(iter+1,:);   % for new mpp approach
   fval=Z(iter+1);
   x0 = X(:,iter+1);%Changing the starting point to the optimal solution obtained in the previous cycle
   iter = iter+1;

   % With the solution found above find the corresponding MPP point
   % using the row vector y, as x was in form of column vector
   f2_mpp(iter,:) = getmpp_pma(R_mpp1,y(iter,:));
   R_mpp1=f2_mpp(iter,:);
   f3_mpp(iter,:)= getmppSD_pma(SD_mpp1,y(iter,:));
   SD_mpp1=f3_mpp(iter,:);
   % perform optimization again
   [errmsg,Z(iter+1),X(:,iter+1),t,c,fail]=bnb_ml60(fun,x0,xstat,xl,xu,a,b,aeq,beq,nonlc,setts,opts);
   disp('solution:'), X
```

```
    y(iter+1,:)=X(:,iter+1).'; % getting row vector corresponding to column vector x obtained above
    %y1=y(iter+1,:);
    r1=0;
    r2=0;
    ri1=0;
    r12=0;
    for i = 1 :9
      r2 = r2+ y(iter+1,i)*stk(i)*(1+ MeanR(i));
      r1 = r1 + y(iter+1,i)*stk(i)*(1+SD_mpp1(i));
    end
    for i = 1 :9
       for j = 1 :9
          ri1 = ri1 + y(iter+1,i)*y(iter+1,j)*stk(i)*stk(j)*CorrC(i,j)*SD_mpp1(i+9)*SD_mpp1(j+9);
          ri2 = ri2 + y(iter+1,i)*y(iter+1,j)*stk(i)*stk(j)*CorrC(i,j)*SigmaR(i)*SigmaR(j);
       end
    end
    ExpValueofPortfolio(1,iter+1)= r1;
    RiskofPortfolio(1,iter+1)= sqrt(ri1);
    ExpValueofPortfolio_MeanR(1,iter+1)= r2;
    Risk_SigmaR(1,iter+1)=sqrt(ri2);

end
```

Code 2: Objective Function objective.m being called by optimize_main.m

---

```
% the objective function('objective.m')
function f = objective_case1(x)
global MeanR MeanN MeanER MeanRmpp MeanRC SigmaR SigmaN SigmaER CovR CovN CovER CorrC SM1 SM2 stk R_mpp1 R_mpp2
a=0;
% Return
for i = 1:9
   a = a + x(i)*MeanER(i);
end
f = -1*a;  % Maximize (returns)
```

Code 3: Constraint Function confun.m being called by optimize_main.m

```
% contraint functions('confun.m')
function [cin,ceq] = confun_case1(x)
global MeanR MeanN MeanER MeanRmpp MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC R_mpp1 SD_mpp1 fp1
f12 f1 f2
f0 = 0;
f1 = 0;
f12 =0;
fp1 = 0;
z= -2.88;
r = 0;
for i = 1 :9
   f0 = f0 + x(i)*stk(i);% Total Investment
   f1 = f1+ x(i)*stk(i)*(1+ MeanR(i));
   f2 = f2+ x(i)*stk(i)*(1+ R_mpp1(i));
   fp1 = fp1 + x(i)*stk(i)*(1+SD_mpp1(i)); % second probabilistic constraint first part
end
% summation of variance involved in the second probabilistic constraint
for i = 1 :9
   for j = 1 :9
      f12 = f12 + x(i)*x(j)*stk(i)*stk(j)*CorrC(i,j)*SD_mpp1(i+9)*SD_mpp1(j+9);
   end
end
ceq = [];
cin = [-fp1+z*sqrt(f12)+f0*(1-0.05)];% final expression of the second probabilictic constraint
f0
```

Code 4: MPP calculation Code getmpp_pma.m being called by optimize_main.m

```
% The code is to get the mpp point corresponding to first prob constraint
function r_mpp = getmpp_pma(mpppt,x)
global MeanR MeanN MeanER MeanRmpp MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC R_mpp1 R_mpp2
optimvalue
%------------------ Rosenblatt Transformation ----------------------------%
u1 = RTrans_case1(mpppt(1:9),MeanER,CovER);
mpppt(1:9)
```

```
u1
%-------------------------    End   ----------------------------------%
%---------------------- MPP Calculation ---------------------------------%
settings=[];
options = optimset('MaxFunEvals',10000,'MaxSQPIter',1000);
a = u1;
xlb=[.00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001];
xub=[.99 .99 .99 .99 .99 .99 .99 .99 .99];
i=2;
x0 = a;
% Performing Optimization for PMA
[u1(i,:),Z(i),exitflag] = fmincon('objective_mpp',x0,[],[],[],[],xlb,xub,'confun_mpp',options);
%-------------------------    End   ----------------------------------%
%----------------- Inverse Rosenblatt Transformation ---------------------%
r_mpp = RTransInv_case1(u1(i,1:9),MeanER,CovER);
u1(i,1:9)
r_mpp
%-------------------------    End   ----------------------------------%
```

Code 5: Objective Function objective_mpp.m being called by getmpp_pma.m

```
Mpp1 Objective function
function p = objective_mpp(u1)
global MeanR MeanN MeanER MeanRmpp MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC R_mpp1 R_mpp2
optimvalue q
p=0;
q=0;
for i = 1:9
      q = q + optimvalue(i)*stk(i)*(1+u1(i)); % first probabilistic constraint
end
p=q;
disp(q);
```

Code 6: Constraint Function confun_mpp.m being called by getmpp_pma.m

```
% Mpp called constraint function
function [cin,ceq] = confun_mpp(u1)
```

```
global MeanR MeanN MeanER MeanRmpp MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC R_mpp1  R_mpp2
SD_mpp1 optimvalue
beta = 1.95;
sum = 0;
   for i = 1:9
      sum = sum + u1(i)*u1(i);
   end
   c1 = sqrt(sum);
ceq = [c1-beta];cin = [];
```

Code 7: RTrans_case1.m called by getmpp_pma.m

```
---------------------------------------------------------------------
%The following Code calcultes the Rosenbat Transformation
function U = RTrans_case1(x,Mean,Cov)
%----------------------- Input values ----------------------------------%
r = 9; %no of scrips
Result_Vector =[];
initial_weight = x;
%-------------------------   End   ---------------------------------%
%-------------------  Start of Transformation  -------------------------%
Result_Vector(1)= (initial_weight(1)-Mean(1))/sqrt(Cov(1,1));
  for k = 2:r
      matrix(1:k,1:k)= Cov(1:k,1:k);
      numer1 =0;
      for j= 1:(k-1)
         numer1 = numer1 + ((cofactor(matrix,k,j)/cofactor(matrix,k,k))*(initial_weight(j)-Mean(j)));
      end
      numer = (initial_weight(k)-Mean(k))+ numer1;
      denom = sqrt(det(matrix)/cofactor(matrix,k,k));
      Result_Vector(k)= (numer/denom);
   end

   for i=1:9
     U(i) = cdf('Normal',Result_Vector(i),Mean(i),sqrt(Cov(i,i)));
   end
%--------------------  End of Transformation  -------------------------%
```

Code 8: RTransInv_case1.m called by getmpp_pma.m

```
%The following Code calcultes Inverse Rosenbat Transformation
function X = RTransInv_case1(u,Mean,Cov)
%------------------------ Input values ----------------------------------%
r = 9; %no of scrips
%--------------------------   End   ----------------------------------%
%-----------------  Start of  Inverse Transformation  ------------------%
X(1) = norminv(u(1),0,1)*sqrt(Cov(1,1))+Mean(1);
   for k = 2:r
      matrix(1:k,1:k)= Cov(1:k,1:k);
      numer1 =0;
      for j= 1:(k-1)
         numer1 = numer1 + ((cofactor(matrix,k,j)/cofactor(matrix,k,k))*(u(j)-Mean(j)));
      end
      denom = sqrt(det(matrix)/cofactor(matrix,k,k));
      X(k)= norminv(u(k),0,1)*denom - numer1 + Mean(k);
   end
%------------------  End of  Inverse Transformation  ------------------%
```

Code 9: MPP calculation Code getmppSD_pma.m being called by optimize_main.m

```
% The code is to get the mpp point corresponding to second prob constraint
function SD_mpp = getmppSD_pma(mpppt1,x1)
global MeanR MeanN MeanER MeanRmpp MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC R_mpp1 SD_mpp1
optimvalue
%------------------ Rosenblatt Transformation ----------------------------%
s1(1:9)  = RTrans_case1(mpppt1(1:9),MeanER,CovER);
s1(10:18) = RTrans_case1(mpppt1(10:18),MeanSD,CovSD);
mpppt1(1:18)
s1
%--------------------------   End   ----------------------------------%
%----------------------- MPP Calculation ---------------------------------%
settings=[];
options = optimset('MaxFunEvals',10000,'MaxSQPIter',1000);
a = s1;
```

```
xlb=[.00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001];
xub=[.99 .99 .99 .99 .99 .99 .99 .99 .99 .99 .99 .99 .99 .99 .99 .99 .99];
i=2;
x0 = a;
[s1(i,:),Z1(i),exitflag] = fmincon('objective_mpp_SD',x0,[],[],[],[],xlb,xub,'confun_mpp_SD',options);
%-------------------------    End   ----------------------------------%
%----------------- Inverse Rosenblatt Transformation ----------------%
SD_mpp(1:9) = RTransInv_case1(s1(i,1:9),MeanER,CovER);
SD_mpp(10:18) = RTransInv_case1(s1(i,10:18),MeanSD,CovSD);
s1(i,1:18)
SD_mpp
%-------------------------    End   ----------------------------------%
```

Code 10: Objective Function objective_mpp_SD.m being called by getmppSD_pma.m

---

```
% Objective function for PMA for Second Probabilistic constraint
function p = objective_mpp_SD(s1)
global MeanR MeanN MeanER MeanRmpp MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrC CorrSD SM1 SM2 stk R_mpp1
SD_mpp1 optimvalue q
p=0;
q0=0;
q1=0;
q2=0;
for i = 1:9
      q0 = q0 + optimvalue(i)*stk(i)*(1-0.05);
end
for i = 1:9
      q1 = q1 + optimvalue(i)*stk(i)*(1+s1(i));
end
for i = 1:9
   for j = 1:9
      q2 = q2 + optimvalue(i)*optimvalue(j)*stk(i)*stk(j)*CorrSD(i,j)*s1(i+9)*s1(j+9);
   end
end
q = q1+2.88*sqrt(q2)-q0;
p=q;
disp(q);
```

Code 11: Constraint Function confun_mpp_SD.m being called by getmppSD_pma.m

---

```
%Constraint function for MPP calculation for second probabilistic constraint
function [cin,ceq] = confun_mpp_SD(s1)
global MeanR MeanN MeanER MeanRmpp MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC R_mpp1 SD_mpp1
optimvalue
beta = 1.95;
c2   = 0;
sum  = 0;
   for i = 1:18
      sum = sum + s1(i)*s1(i);
   end
c2   = sqrt(sum);
ceq  = [c2-beta];
cin  = [];
```

Model IV

Code 1: optimize_main.m % This is the main code which calls the other functions and codes.

```
% The code is to perform the optimization using SORA for Model 4.
% MeanER contains the mean of  expected returns from bootstrap
% SigmaER contains the standard deviation of expected returns from bootstrap
% CovER contains the covariance matrix for expected returns from bootstrap
% MeanN contains the mean of the weights.
% SigmaN contains the standard deviation of weights.
% CovN contains the covariance matrix for weights.
% MeanR contains the mean of returns
% SigmaR contains the standard deviation of returns
% CovR contains the covariance matrix for returns
% CorrC is the correlation matrix from the Cov matrix of returns
% % xu and xl are vector containing upper and lower bound
% Stk contains the last traded price of the scripts
```

```
% confun.m contains the nonlinear constraints
% objective.m contains the objective function
clc;clear all;
global MeanR MeanN MeanER MeanRmpp MeanRC SigmaR SigmaN SigmaER CovR CovN CovER CorrC R_mpp1 optimvalue q
%-------------------------------------------------------------------------%
%             Input parameters to read from Xls file                      %
%-------------------------------------------------------------------------%
filename = 'data.xls'; %the xls filename
%----------------------------  for returns ------------------------------%
sheet = 'InputReturns';      % The name of the sheet in the file above
[CovR, header] = xlsread(filename, sheet, 'A1:J10');
[MeanR] = xlsread(filename, sheet, 'B14:J14');
for i=1:9
   for j=1:i
      CovR(i,j)=CovR(j,i);
   end
end
[SigmaR, CorrC] = cov2corr(CovR);
[stk] = xlsread(filename, sheet, 'B21:J21');
%----------------------------  for weights ------------------------------ %
sheet = 'InputWeights';      % The name of the sheet in the file above
[CovN, header] = xlsread(filename, sheet, 'A1:J10');
[MeanN] = xlsread(filename, sheet, 'B14:J14');
for i=1:9
   for j=1:i
      CovN(i,j)=CovN(j,i);
   end
end
[SigmaN, CorrN] = cov2corr(CovN);
%----------------------- for expected returns from bootstrap ------------------------------%
filename = 'bootstrapdata_nine_scrips.xls'; %the xls filename
sheet = 'InpExpreturns';      % The name of the sheet in the file above
[dataER, header] = xlsread(filename, sheet, 'A2:I502');
CovER = cov(dataER);
MeanER = mean(dataER);
[SigmaER, CorrER] = cov2corr(CovER);
%----------------------- for standard deviations from bootstrap ------------------------------%
filename = 'bootstrapdata_nine_scrips.xls'; %the xls filename
```

```
sheet = 'InpSDreturns';      % The name of the sheet in the file above
[dataSD, header] = xlsread(filename, sheet, 'K2:S502');
CovSD = cov(dataSD);
MeanSD = mean(dataSD);
[SigmaSD, CorrSD] = cov2corr(CovSD);
%------------------ Starting the loop to vary initial amount----------
%%v0=1000000;
%for g = 1:30
%   v0=v0+10000;
%-----------------  END of Inputs  from XLs sheet ------------------------%
R_mpp1 = MeanR;
%----------------------NEWBNB-------------------------------------------%
fun        = 'objective_case1';
x0         = [1500 1500 1500 1500 1500 1500 1500 1500 1500 0]';
xstat      = [1 1 1 1 1 1 1 1 1 1]';
xl         = [0 0 0 0 0 0 0 0 0 0]';
xu         = [5000 5000 5000 5000 5000 5000 5000 5000 5000 2000000]';
v0=1000000;
v1=.3*v0;
b          = [v0 0 v1 v1 v1 v1 v1 v1 v1 v1]';
a(1,1:10)=[76.80 45.07 107.20 110.70 110.08 21.78 35.19 13.68 84.72 0];
a(2,1:10)=     [-1*stk(1)*(1+R_mpp1(1))     -1*stk(2)*(1+R_mpp1(2))     -1*stk(3)*(1+R_mpp1(3))     -1*stk(4)*(1+R_mpp1(4))     -1*stk(5)*(1+R_mpp1(5))     -
1*stk(6)*(1+R_mpp1(6)) -1*stk(7)*(1+R_mpp1(7)) -1*stk(8)*(1+R_mpp1(8)) -1*stk(9)*(1+R_mpp1(9)) 1];
a(3,1:10)   = [76.8 0 0 0 0 0 0 0 0 0];
a(4,1:10)   = [ 0 45.07 0 0 0 0 0 0 0 0];
a(5,1:10)   = [0 0 107.2 0 0 0 0 0 0 0];
a(6,1:10)   = [0 0 0 110.7 0 0 0 0 0 0];
a(7,1:10)   = [0 0 0 0 110.08 0 0 0 0 0];
a(8,1:10)   = [0 0 0 0 0 21.78 0 0 0 0];
a(9,1:10)   = [0 0 0 0 0 0 35.19 0 0 0];
a(10,1:10)   = [0 0 0 0 0 0 0 13.68 0 0];
a(11,1:10)   = [0 0 0 0 0 0 0 0 84.72 0];
aeq        = [];
beq        = [];
nonlc      = 'confun_case1';
setts      = [];
opts       = optimset('display','off','MaxSQPIter',1000);
%--------------------------------------------------------------------------%
```

```
%                OPTIMIZATION PART STARTS HERE                    %
%-------------------------------------------------------------------------%
% x0 is the starting point of optimization.
iter = 0;
fval=0;
%Optimaization, BNB being called
[errmsg,Z(iter+1),X(:,iter+1),t,c,fail]=bnb_ml60(fun,x0,xstat,xl,xu,a,b,aeq,beq,nonlc,setts,opts);
disp('solution:'), X
y(iter+1,:)=X(:,iter+1).'; % getting row vector corresponding to column vector x obtained above
r1=0;
r2=0;
ri2=0;
for i = 1 :9
   r2 = r2+ y(iter+1,i)*stk(i)*(1+ MeanR(i));  %ExpectedValueofPortfolio
   r1 = r1 + y(iter+1,i)*stk(i)*(1+R_mpp1(i)); %ExpectedValueofPortfolio
end
for i = 1 :9
   for j = 1 :9
      ri2 = ri2 + y(iter+1,i)*y(iter+1,j)*stk(i)*stk(j)*CorrC(i,j)*SigmaR(i)*SigmaR(j);
   end
end
ExpValueofPortfolio(1,iter+1)= r1;
ExpValueofPortfolio_MeanR(1,iter+1)= r2;
Risk_SigmaR(1,iter+1)=sqrt(ri2);
while((abs(Z(iter+1)-fval)>0))
%while(iter<=4)
   optimvalue = y(iter+1,:);   % for new mpp approach
   fval=Z(iter+1);
   x0 = X(:,iter+1);%Changing the starting point to the optimal solution obtained in the previous cycle
   iter = iter+1;
   % With the solution found above find the corresponding MPP point
   % using the row vector y, as x was in form of column vector
   f2_mpp(iter,:) = getmpp_pma(R_mpp1,y(iter,:));
   R_mpp1=f2_mpp(iter,:);
   % perform optimization again
   [errmsg,Z(iter+1),X(:,iter+1),t,c,fail]=bnb_ml60(fun,x0,xstat,xl,xu,a,b,aeq,beq,nonlc,setts,opts);
   disp('solution:'), X
   y(iter+1,:)=X(:,iter+1).'; % getting row vector corresponding to column vector x obtained above
```

```
    r1=0;
    r2=0;
    ri2=0;
    for i = 1 :9
        r2 = r2+ y(iter+1,i)*stk(i)*(1+ MeanR(i));  %ExpectedValueofPortfolio
        r1 = r1 + y(iter+1,i)*stk(i)*(1+R_mpp1(i)); %ExpectedValueofPortfolio
    end
    for i = 1 :9
        for j = 1 :9
            ri2 = ri2 + y(iter+1,i)*y(iter+1,j)*stk(i)*stk(j)*CorrC(i,j)*SigmaR(i)*SigmaR(j);
        end
    end
    ExpValueofPortfolio(1,iter+1)= r1;
    ExpValueofPortfolio_MeanR(1,iter+1)= r2;
    Risk_SigmaR(1,iter+1)=sqrt(ri2);
end
%---------------------------------------------------------------------------%
%                        OPTIMIZATION ENDS                                  %
%---------------------------------------------------------------------------%
```

Code 2: Objective Function objective.m being called by optimize_main.m

---

```
% the objective function('objective.m')
% the objective function('objective.m')
function f = objective_case1(x)
global MeanR MeanN MeanER MeanRmpp MeanRC SigmaR SigmaN SigmaER CovR CovN CovER CorrC SM1 SM2 stk R_mpp1
% RC
z = x(10);
f = -1*(z);  %Maximize rc
```

Code 3: Constraint Function confun.m being called by optimize_main.m

---

```
% contraint functions('confun.m')
function [cin,ceq] = confun_case1(x)
global MeanR MeanN MeanER MeanRmpp MeanRC SigmaR SigmaN SigmaER CovR CovN CovER CorrC SM1 SM2 stk R_mpp1
f0 = 0;
```

```
f1 = 0;
f12 =0;
z= -2.88;%corresponding to alpha_var=.02
for i = 1 :9
   f0 = f0 + x(i)*stk(i);% Total Investment
   f1 = f1+ x(i)*stk(i)*(1+ MeanR(i));
end
% summation of variance
for i = 1 :9
   for j = 1 :9
      f12 = f12 + x(i)*x(j)*stk(i)*stk(j)*CorrC(i,j)*SigmaR(i)*SigmaR(j);
   end
end
ceq = [];
cin = [-f1+z*sqrt(f12)+f0*(1-0.05)];
f0
```

Code 4: MPP calculation Code getmpp_pma.m being called by optimize_main.m

---

```
% The code is to get the mpp point corresponding to first prob constraint
function r_mpp = getmpp_pma(mpppt,x)
global MeanR MeanN MeanER MeanRmpp MeanRC SigmaR SigmaN SigmaER CovR CovN CovER CorrC SM1 SM2 stk R_mpp1 optimvalue
%----------------------- Input values ----------------------------------%
%------------------ Rosenblatt Transformation ---------------------------%
u1 = RTrans_case1(mpppt(1:9),MeanER,CovER);
mpppt(1:9)
u1
%-------------------------   End   ----------------------------------%
%----------------------- MPP Calculation ---------------------------------%
settings=[];
options = optimset('MaxFunEvals',10000,'MaxSQPIter',1000);
a = u1;
xlb=[.00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001];
xub=[.99 .99 .99 .99 .99 .99 .99 .99 .99];
i=2;
x0 = a;
[u1(i,:),Z(i),exitflag] = fmincon('objective_mpp',x0,[],[],[],[],xlb,xub,'confun_mpp',options);
```

```
%-------------------------    End   ----------------------------------%
%----------------- Inverse Rosenblatt Transformation ----------------%
%u(i,:)=a(:,i).'; % converting back to row
r_mpp = RTransInv_case1(u1(i,1:9),MeanER,CovER);
u1(i,1:9)
r_mpp
%-------------------------    End   ----------------------------------%
```

Code 5: Objective Function objective_mpp.m being called by getmpp_pma.m

```
% Objective function for MPP calculation
function p = objective_mpp(u1)
global MeanR MeanN MeanER MeanRmpp MeanRC SigmaR SigmaN SigmaER CovR CovN CovER CorrC SM1 SM2 stk R_mpp1 optimvalue q
p=0;
q=0;
for i = 1:9
      q = q + optimvalue(i)*stk(i)*(1+u1(i)); % first probabilistic constraint
end
p=q;
disp(q);
```

Code 6: Constraint Function confun_mpp.m being called by getmpp_pma.m

```
function [cin,ceq] = confun_mpp(u1)
global MeanR MeanN MeanER MeanRmpp MeanRC SigmaR SigmaN SigmaER CovR CovN CovER CorrC R_mpp1 optimvalue
beta = 1.95;
sum = 0;
   for i = 1:9
      sum = sum + u1(i)*u1(i);
   end
   c1 = sqrt(sum);
ceq = [c1-beta];
cin = [];
```

Code 7: RTrans_case1.m called by getmpp_pma.m

```
%The following Code calcultes the Rosenbat Transformation
function U = RTrans_case1(x,Mean,Cov)
%------------------------ Input values ----------------------------------%
r = 9; %no of scrips
Result_Vector =[];
initial_weight = x;
%--------------------------   End   ----------------------------------%
%--------------------  Start of Transformation  -------------------------%
Result_Vector(1)= (initial_weight(1)-Mean(1))/sqrt(Cov(1,1));
  for k = 2:r
      matrix(1:k,1:k)= Cov(1:k,1:k);
      numer1 =0;
      for j= 1:(k-1)
         numer1 = numer1 + ((cofactor(matrix,k,j)/cofactor(matrix,k,k))*(initial_weight(j)-Mean(j)));
      end
      numer = (initial_weight(k)-Mean(k))+ numer1;
      denom = sqrt(det(matrix)/cofactor(matrix,k,k));
      Result_Vector(k)= (numer/denom);
   end

   for i=1:9
     U(i) = cdf('Normal',Result_Vector(i),Mean(i),sqrt(Cov(i,i)));
   end
%---------------------  End of Transformation  -------------------------%
```

Code 8: RTransInv_case1.m called by getmpp_pma.m

```
%The following Code calcultes Inverse Rosenbat Transformation
function X = RTransInv_case1(u,Mean,Cov)
%------------------------ Input values ----------------------------------%
r = 9; %no of scrips
%--------------------------   End   ----------------------------------%
%-----------------  Start of  Inverse Transformation  -------------------%
X(1) = norminv(u(1),0,1)*sqrt(Cov(1,1))+Mean(1);
   for k = 2:r
```

```
        matrix(1:k,1:k)= Cov(1:k,1:k);
        numer1 =0;
        for j= 1:(k-1)
            numer1 = numer1 + ((cofactor(matrix,k,j)/cofactor(matrix,k,k))*(u(j)-Mean(j)));
        end
        denom = sqrt(det(matrix)/cofactor(matrix,k,k));
        X(k)= norminv(u(k),0,1)*denom - numer1 + Mean(k);
    end
%------------------  End of  Inverse Transformation  ----------------------%
```

Model 5

Code 1 : optimize_main.m

```
% The code is to perform the optimization using SORA for Model 5.
% MeanER contains the mean of  expected returns from bootstrap
% SigmaER contains the standard deviation of expected returns from bootstrap
% CovER contains the covariance matrix for expected returns from bootstrap
% MeanN contains the mean of the weights.
% SigmaN contains the standard deviation of weights.
% CovN contains the covariance matrix for weights.
% MeanR contains the mean of returns
% SigmaR contains the standard deviation of returns
% CovR contains the covariance matrix for returns
% CorrC is the correlation matrix from the Cov matrix of returns
% xu and xl are vector containing upper and lower bound
% Stk contains the last traded price of the scripts
% confun.m contains the nonlinear constraints
% objective.m contains the objective function
% R_mpp1 and SD_mpp1 contain the mpp values corresponding to first and second probabilistic constraints
clc;clear all;
global MeanR MeanN MeanER MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC stk R_mpp1 SD_mpp1 optimvalue q
 %-------------------------------------------------------------------------%
%            Input parameters to read from Xls file                       %
%-------------------------------------------------------------------------%
```

```
 filename = 'data.xls'; %the xls filename
 %----------------------------  for returns ------------------------------%
 sheet = 'InputReturns';      % The name of the sheet in the file above
[CovR, header] = xlsread(filename, sheet, 'A1:J10');
[MeanR] = xlsread(filename, sheet, 'B14:J14');
for i=1:9
    for j=1:i
        CovR(i,j)=CovR(j,i);
    end
end
[SigmaR, CorrC] = cov2corr(CovR);
[stk] = xlsread(filename, sheet, 'B21:J21');
 %----------------------------  for weights ------------------------------ %
 sheet = 'InputWeights';      % The name of the sheet in the file above
[CovN, header] = xlsread(filename, sheet, 'A1:J10');
[MeanN] = xlsread(filename, sheet, 'B14:J14');
for i=1:9
    for j=1:i
        CovN(i,j)=CovN(j,i);
    end
end
[SigmaN, CorrN] = cov2corr(CovN);
 %----------------------- for expected returns from bootstrap ------------------------------%
filename = 'bootstrapdata_nine_scrips.xls'; %the xls filename
sheet = 'InpExpreturns';      % The name of the sheet in the file above
[dataER, header] = xlsread(filename, sheet, 'A2:I502');
CovER = cov(dataER);
MeanER = mean(dataER);
[SigmaER, CorrER] = cov2corr(CovER);
%----------------------- for standard deviations from bootstrap ------------------------------%
filename = 'bootstrapdata_nine_scrips.xls'; %the xls filename
sheet = 'InpSDreturns';      % The name of the sheet in the file above
[dataSD, header] = xlsread(filename, sheet, 'K2:S502');
CovSD = cov(dataSD);
MeanSD = mean(dataSD);
[SigmaSD, CorrSD] = cov2corr(CovSD);
%-----------------  END of Inputs  from XLs sheet ------------------------%
R_mpp1 = MeanR;
```

```
SD_mpp1 = cat(2,MeanR,MeanSD);
%----------------------NEWBNB-------------------------------------------%
fun        = 'objective_case1';
x0         = [1500 1500 1500 1500 1500 1500 1500 1500 1500 0]';
xstat      = [1 1 1 1 1 1 1 1 1 1]';
xl         = [0 0 0 0 0 0 0 0 0 0]';
xu         = [5000 5000 5000 5000 5000 5000 5000 5000 5000 2000000]';
v0=100000;
v1=.3*v0;
b          = [v0 0 v1 v1 v1 v1 v1 v1 v1 v1]';
a(1,1:10)=[76.80 45.07 107.20 110.70 110.08 21.78 35.19 13.68 84.72 0];
a(2,1:10)=   [-1*stk(1)*(1+R_mpp1(1))     -1*stk(2)*(1+R_mpp1(2))     -1*stk(3)*(1+R_mpp1(3))     -1*stk(4)*(1+R_mpp1(4))     -1*stk(5)*(1+R_mpp1(5))     -
1*stk(6)*(1+R_mpp1(6))  -1*stk(7)*(1+R_mpp1(7))  -1*stk(8)*(1+R_mpp1(8))  -1*stk(9)*(1+R_mpp1(9))  1];
a(3,1:10)   = [76.8 0 0 0 0 0 0 0 0 0];
a(4,1:10)   = [ 0 45.07 0 0 0 0 0 0 0 0];
a(5,1:10)   = [0 0 107.2 0 0 0 0 0 0 0];
a(6,1:10)   = [0 0 0 110.7 0 0 0 0 0 0];
a(7,1:10)   = [0 0 0 0 110.08 0 0 0 0 0];
a(8,1:10)   = [0 0 0 0 0 21.78 0 0 0 0];
a(9,1:10)   = [0 0 0 0 0 0 35.19 0 0 0];
a(10,1:10)   = [0 0 0 0 0 0 0 13.68 0 0];
a(11,1:10)   = [0 0 0 0 0 0 0 0 84.72 0];
aeq        = [];
beq        = [];
nonlc      = 'confun_case1';
setts      = [];
opts       = optimset('display','off','MaxSQPIter',1000);
%------------------------------------------------------------------------%
%                 OPTIMIZATION PART STARTS HERE
%------------------------------------------------------------------------%
iter = 0;
fval=0;
% Optimization, BNB being called
[errmsg,Z(iter+1),X(:,iter+1),t,c,fail]=bnb_ml60(fun,x0,xstat,xl,xu,a,b,aeq,beq,nonlc,setts,opts);
disp('solution:'), X
y(iter+1,:)=X(:,iter+1).'; % getting row vector corresponding to column vector x obtained above
 r1=0;
r2=0;
```

```
ri1=0;
ri2=0;
for i = 1 :9
   r2 = r2+ y(iter+1,i)*stk(i)*(1+ MeanR(i));%ExpectedValueofPortfolio
   r1 = r1 + y(iter+1,i)*stk(i)*(1+SD_mpp1(i)); %ExpectedValueofPortfolio
end
for i = 1 :9
   for j = 1 :9
      ri1 = ri1 + y(iter+1,i)*y(iter+1,j)*stk(i)*stk(j)*CorrC(i,j)*SD_mpp1(i+9)*SD_mpp1(j+9);
      ri2 = ri2 + y(iter+1,i)*y(iter+1,j)*stk(i)*stk(j)*CorrC(i,j)*SigmaR(i)*SigmaR(j);
   end
end
ExpValueofPortfolio(1,iter+1)= r1;
RiskofPortfolio(1,iter+1)= sqrt(ri1);
ExpValueofPortfolio_MeanR(1,iter+1)= r2;
Risk_SigmaR(1,iter+1)=sqrt(ri2);
 while((abs(Z(iter+1)-fval)>.01))
   optimvalue = y(iter+1,:);   % for new mpp approach
   fval=Z(iter+1);
   x0 = X(:,iter+1);%Changing the starting point to the optimal solution obtained in the previous
   cycle
   iter = iter+1;
   % With the solution found above find the corresponding MPP point
   % using the row vector y, as x was in form of column vector
   f2_mpp(iter,:) = getmpp_pma(R_mpp1,y(iter,:));    %PMA code being called for MPP
   R_mpp1=f2_mpp(iter,:);
   f3_mpp(iter,:)= getmppSD_pma(SD_mpp1,y(iter,:)); %PMA code being called for MPP
   SD_mpp1=f3_mpp(iter,:);
   % perform optimization again
   [errmsg,Z(iter+1),X(:,iter+1),t,c,fail]=bnb_ml60(fun,x0,xstat,xl,xu,a,b,aeq,beq,nonlc,setts,opts);
   disp('solution:'), X
   y(iter+1,:)=X(:,iter+1).'; % getting row vector corresponding to column vector x obtained above
   r1=0;
   r2=0;
   ri1=0;
   ri2=0;
   for i = 1 :9
     r2 = r2+ y(iter+1,i)*stk(i)*(1+ MeanR(i));
```

```
    r1 = r1 + y(iter+1,i)*stk(i)*(1+SD_mpp1(i));
  end
  for i = 1 :9
    for j = 1 :9
      ri1 = ri1 + y(iter+1,i)*y(iter+1,j)*stk(i)*stk(j)*CorrC(i,j)*SD_mpp1(i+9)*SD_mpp1(j+9);
      ri2 = ri2 + y(iter+1,i)*y(iter+1,j)*stk(i)*stk(j)*CorrC(i,j)*SigmaR(i)*SigmaR(j);
    end
  end
  ExpValueofPortfolio(1,iter+1)= r1;
  RiskofPortfolio(1,iter+1)= sqrt(ri1);
  ExpValueofPortfolio_MeanR(1,iter+1)= r2;
  Risk_SigmaR(1,iter+1)=sqrt(ri2);
end
%-------------------------------------------------------------------------%
%                        OPTIMIZATION ENDS                                %
%-------------------------------------------------------------------------%
```

Code 2: Objective Function objective.m being called by optimize_main.m

```
% the objective function('objective.m')
function f = objective_case1(x)
global MeanR MeanN MeanER MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC stk R_mpp1 SD_mpp1
a=0;
% RC
z = x(10);
f = -1*z;
```

Code 3: Constraint Function confun.m being called by optimize_main.m

```
% contraint functions('confun.m')
function [cin,ceq] = confun_case1(x)
global MeanR MeanN MeanER MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC stk R_mpp1 SD_mpp1
f0 = 0;
f1 = 0;
f12 =0;
```

```
fp1 = 0;
z= -2.88;  %corresponding to alpha_VaR
r = x(10);
for i = 1 :9
   f0 = f0 + x(i)*stk(i);% Total Investment
   f1 = f1+ x(i)*stk(i)*(1+ MeanR(i));
   fp1 = fp1 + x(i)*stk(i)*(1+SD_mpp1(i)); % second probabilistic constraint first part
end
% summation of variance involved in the second probabilistic constraint
for i = 1 :9
   for j = 1 :9
      f12 = f12 +
         x(i)*x(j)*stk(i)*stk(j)*CorrC(i,j)*SD_mpp1(i+9)*SD_mpp1(j+9);
   end
end
ceq = [];
cin = [-fp1+z*sqrt(f12)+f0*(1-0.05)];
```

Code 4: MPP calculation Code getmpp_pma.m being called by optimize_main.m

---

```
%The code is to get the mpp point corresponding to first prob. constraint
function r_mpp = getmpp_pma(mpppt,x)
global MeanR MeanN MeanER MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC stk R_mpp1 SD_mpp1
%------------------ Rosenblatt Transformation --------------------------%
u1 = RTrans_case1(mpppt(1:9),MeanER,CovER); % Calling RTrans_case1.m
mpppt(1:9)
u1
%--------------------------   End   ----------------------------------%
%----------------------- MPP Calculation -------------------------------%
settings=[];
options = optimset('MaxFunEvals',10000,'MaxSQPIter',1000);
a = u1;
xlb=[.00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001];
xub=[.99 .99 .99 .99 .99 .99 .99 .99 .99];
i=2;
```

```
x0 = a;
% Optimization function called now
[u1(i,:),Z(i),exitflag] = fmincon('objective_mpp',x0,[],[],[],[],xlb,xub,'confun_mpp',options);
%-------------------------    End   ----------------------------------%
%----------------- Inverse Rosenblatt Transformation ------------------%
r_mpp = RTransInv_case1(u1(i,1:9,MeanER,CovER);%Calling RTransInv_case1.m
u1(i,1:9)
r_mpp
%-------------------------    End   ----------------------------------%
```

Code 5: Objective Function objective_mpp.m being called by getmpp_pma.m

```
% Objective function for MPP calculation for first prob. constraint
function p = objective_mpp(u1)
global MeanR MeanN MeanER MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC stk R_mpp1 SD_mpp1
q
p=0;
q=0;
for i = 1:9
   q = q + optimvalue(i)*stk(i)*(1+u1(i)); % first probabilistic constraint
end
p=q;
disp(q);
```

Code 6: Constraint Function confun_mpp.m being called by getmpp_pma.m

```
% Constraint function for MPP calculation for first prob. constraint
function [cin,ceq] = confun_mpp(u1)
global MeanR MeanN MeanER MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC stk R_mpp1 SD_mpp1
beta = 1.95;
sum = 0;
   for i = 1:9
      sum = sum + u1(i)*u1(i);
   end
```

```
    c1 = sqrt(sum);
ceq = [c1-beta];
cin = [];
```

Code 7: RTrans_case1.m called by getmpp_pma.m

```
%The following Code does the Rosenbat Transformation
function U = RTrans_case1(x,Mean,Cov)
%------------------------ Input values ----------------------------------%
r = 9; %no of scrips
Result_Vector =[];
initial_weight = x;
%--------------------------   End   ----------------------------------%
%--------------------  Start of Transformation  ------------------------%
Result_Vector(1)= (initial_weight(1)-Mean(1))/sqrt(Cov(1,1));
for k = 2:r
    matrix(1:k,1:k)= Cov(1:k,1:k);
    numer1 =0;
    for j= 1:(k-1)
          numer1 = numer1 +
       ((cofactor(matrix,k,j)/cofactor(matrix,k,k))*(initial_weight(j)-
         Mean(j)));
    end
    numer = (initial_weight(k)-Mean(k))+ numer1;
    denom = sqrt(det(matrix)/cofactor(matrix,k,k));
    Result_Vector(k)= (numer/denom);
end
for i=1:9
    U(i) = cdf('Normal',Result_Vector(i),Mean(i),sqrt(Cov(i,i)));
end
%----------------------  End of Transformation  -------------------------%
```

Code 8: RTransInv_case1.m called by getmpp_pma.m

```
%The following Code does the Inverse Rosenbat Transformation
function X = RTransInv_case1(u,Mean,Cov)
```

```
%------------------------ Input values ----------------------------------%
r = 9; %no of scrips
%-------------------------   End   ----------------------------------%
%-----------------  Start of  Inverse Transformation  ------------------%
X(1) = norminv(u(1),0,1)*sqrt(Cov(1,1))+Mean(1);
for k = 2:r
      matrix(1:k,1:k)= Cov(1:k,1:k);
      numer1 =0;
      for j= 1:(k-1)
         numer1 = numer1 +
         ((cofactor(matrix,k,j)/cofactor(matrix,k,k))*(u(j)-Mean(j)));
      end
      denom = sqrt(det(matrix)/cofactor(matrix,k,k));
      X(k)= norminv(u(k),0,1)*denom - numer1 + Mean(k);
end
%-----------------  End of  Inverse Transformation  --------- ---------%
```

Code 9: MPP calculation Code getmppSD_pma.m being called by optimize_main.m

---

```
%The code is to get the MPP corresponding to second prob. constraint
function SD_mpp = getmppSD_pma(mpppt1,x1)
global MeanR MeanN MeanER MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC stk R_mpp1 SD_mpp1
%------------------------ Input values ----------------------------------%
%------------------ Rosenblatt Transformation -------------------------%
s1(1:9)  = RTrans_case1(mpppt1(1:9),MeanER,CovER);
s1(10:18) = RTrans_case1(mpppt1(10:18),MeanSD,CovSD);
mpppt1(1:18)
s1
%-------------------------   End   ----------------------------------%
%----------------------- MPP Calculation -------------------------------%
settings=[];
options = optimset('MaxFunEvals',10000,'MaxSQPIter',1000);
a = s1;
xlb=[.00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001 .00001];
xub=[.99 .99 .99 .99 .99 .99 .99 .99 .99 .99 .99 .99 .99 .99 .99 .99 .99 .99];
i=2;
```

```
x0 = a;
%Optimization Process starts
[s1(i,:),Z1(i),exitflag] = fmincon('objective_mpp_SD',x0,[],[],[],[],xlb,xub,'confun_mpp_SD',options);
%--------------------------    End   ----------------------------------%
%----------------- Inverse Rosenblatt Transformation -------------------%
SD_mpp(1:9) = RTransInv_case1(s1(i,1:9),MeanER,CovER);
SD_mpp(10:18) = RTransInv_case1(s1(i,10:18),MeanSD,CovSD);
s1(i,1:18)
SD_mpp
%--------------------------    End   ----------------------------------%
```

Code 10: Objective Function objective_mpp_SD.m being called by getmppSD_pma.m

```
% Objective function for MPP calculation for second prob. constraint
function p = objective_mpp_SD(s1)
global MeanR MeanN MeanER MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC stk R_mpp1 SD_mpp1
q
p=0;
q0=0;
q1=0;
q2=0;
for i = 1:9
       q0 = q0 + optimvalue(i)*stk(i)*(1-0.05);
end
for i = 1:9
       q1 = q1 + optimvalue(i)*stk(i)*(1+s1(i));
end
for i = 1:9
    for j = 1:9
       q2 = q2 +
  optimvalue(i)*optimvalue(j)*stk(i)*stk(j)*CorrSD(i,j)*s1(i+9)*s1(j+9);
    end
end
q = q1+2.88*sqrt(q2)-q0; % Second Probabilistic constraint
p=q;
disp(q);
```

Code 11: Constraint Function confun_mpp_SD.m being called by getmppSD_pma.m

---

```
% Constraint function for MPP calculation for second prob. constraint
function [cin,ceq] = confun_mpp_SD(s1)
global MeanR MeanN MeanER MeanRC MeanSD SigmaR SigmaN SigmaER CovR CovN CovER CovSD CorrSD CorrC stk R_mpp1 SD_mpp1
beta = 1.95;
c2   = 0;
sum  = 0;
for i = 1:18
   sum = sum + s1(i)*s1(i);
end
c2   = sqrt(sum);
ceq  = [c2-beta];
cin  = [];
```

Branch and Bound Code (BNB)

```
function [errmsg,Z,X,t,c,fail] = BNB20(fun,x0,xstat,xl,xu,a,b,aeq,beq,nonlc,setts,opts,varargin);
% BNB20 Finds the constrained minimum of a function of several possibly integer variables.
% Usage: [errmsg,Z,X,t,c,fail] =
%       BNB20(fun,x0,xstatus,xlb,xub,A,B,Aeq,Beq,nonlcon,settings,options,P1,P2,...)%
% BNB solves problems of the form:
% Minimize F(x) subject to: xlb <= x0 <=xub
%                     A*x <= B  Aeq*x=Beq
%                     C(x)<=0  Ceq(x)=0
%                     x(i) is continuous for xstatus(i)=0
%                     x(i) integer for xstatus(i)= 1
%                     x(i) fixed for xstatus(i)=2
% BNB uses:
% Optimization Toolbox Version 2.0 (R11) 09-Oct-1998
% From this toolbox fmincon.m is called. For more info type help fmincon.%
% fun is the function to be minimized and should return a scalar. F(x)=feval(fun,x).
% x0 is the starting point for x. x0 should be a column vector.%
% xstatus is a column vector describing the status of every variable x(i).
% 0 for continuous variable x(i)
```

```
% 1 for integer variable x(i)
% 2 for fixed variable x(i)%
% xlb and xub are column vectors with lower and upper bounds for x.
% A and Aeq are matrices for the linear constrains.
% B and Beq are column vectors for the linear constrains.
% nonlcon is the function for the nonlinear constrains.
% [C(x);Ceq(x)]=feval(nonlcon,x). Both C(x) and Ceq(x) should be column vectors.%
% errmsg is a string containing an error message if BNB found an error in the input.
% Z is the scalar result of the minimization, X the values of the accompanying variables.
% t is the time elapsed while the algorithm BNB has run and c is the number of BNB cycles.
% fail is the number of nonconvergent leaf sub-problems.%
% settings is a row vector with settings for BNB:
% settings(1) (standard 0) if 1: if the sub-problem does not converge do not branch it and
% raise fail by one. Normally BNB will always branch a nonconvergent sub-problem so it can
% try again to find a solution.
% A sub-problem that is a leaf of the branch-and-bound-tree cannot be branched. If such
% a problem does not converge it will be considered infeasible and fail will be raised by one.
% settings(2) is the handle of main BNB GUI. Leave empty.%
% options is an options structure. For details type help optimset.%
% options.maxSQPiter is a variable used by fmincon (if modified as described in bnb20.m).
% maxSQPiter cannot be set by optimset because it is not part of the standard options
% structure. maxSQPiter is 1000 by default.%
% P1,P2,... are parameters to be passed to fun and nonlcon.
% F(x)=feval(fun,x,P1,P2,...). [C(x);Ceq(x)]=feval(nonlcon,x,P1,P2,...).
% Type edit BNB20 for more info.


% E.C. Kuipers
% e-mail E.C.Kuipers@cpedu.rug.nl
% FI-Lab
% Applied Physics
% Rijksuniversiteit Groningen

global maxSQPiter;
% *** STEP 0 *** CHECKING INPUT
Z=[]; X=[]; t=0; c=0; fail=0;
if nargin<2, errmsg='BNB needs at least 2 input arguments.';
   return;
```

```
end;
if isempty(fun), errmsg='No fun found.';
   return;
elseif ~ischar(fun) & ~isa(fun,'inline')
    errmsg='fun must be a string - hier gaat het dus fout.';
   return;
end;
if isempty(x0)
  errmsg='No x0 found.';
     return;
elseif ~isnumeric(x0) | ~isreal(x0) | size(x0,2)>1
  errmsg='x0 must be a real column vector.';
     return;
end;
xstatus=zeros(size(x0));
if nargin>2 & ~isempty(xstat)
  if isnumeric(xstat) & isreal(xstat) & all(size(xstat)<=size(x0))
     if all(xstat==round(xstat) & 0<=xstat & xstat<=2)
        xstatus(1:size(xstat))=xstat;
     else errmsg='xstatus must consist of the integers 0,1 en 2.';
        return;
     end;
  else errmsg='xstatus must be a real column vector the same size as x0.';
     return;
  end;
end;
xlb=zeros(size(x0));
xlb(find(xstatus==0))=-inf;
if nargin>3 & ~isempty(xl)
  if isnumeric(xl) & isreal(xl) & all(size(xl)<=size(x0))
     xlb(1:size(xl,1))=xl;
  else errmsg='xlb must be a real column vector the same size as x0.';
     return;
  end;
end;
xlb(find(xstatus==2))=x0(find(xstatus==2));
xub=ones(size(x0));
xub(find(xstatus==0))=inf;
```

```
if nargin>4 & ~isempty(xu)
  if isnumeric(xu) & isreal(xu) & all(size(xu)<=size(x0))
    xub(1:size(xu,1))=xu;
  else errmsg='xub must be a real column vector the same size as x0.';
    return;
  end;
end;
if any(x0>xub)
  errmsg='x0 must be in the range x0 <=xub.';
    return;
elseif any(xstatus==1 & (~isfinite(xub) | xub~=round(xub)))
  errmsg='xub(i) must be an integer if x(i) is an integer variabale.';
    return;
end;
xub(find(xstatus==2))=x0(find(xstatus==2));
A=[];
if nargin>5 & ~isempty(a)
  if isnumeric(a) & isreal(a) & size(a,2)==size(x0,1)
    A=a;
  else errmsg='Matrix A not correct.';
    return;
  end;
end;
B=[];
if nargin>6 & ~isempty(b)
  if isnumeric(b) & isreal(b) & all(size(b)==[size(A,1) 1])
  B=b;
  else errmsg='Column vector B not correct.';
    return;
  end;
end;
if isempty(B) & ~isempty(A)
  B=zeros(size(A,1),1);
end;
Aeq=[];
if nargin>7 & ~isempty(aeq)
  if isnumeric(aeq) & isreal(aeq) & size(aeq,2)==size(x0,1)
    Aeq=aeq;
```

```
  else errmsg='Matrix Aeq not correct.';
    return;
  end;
end;
Beq=[];
if nargin>8 & ~isempty(beq)
  if isnumeric(beq) & isreal(beq) & all(size(beq)==[size(Aeq,1) 1])
    Beq=beq;
  else errmsg='Column vector Beq not correct.';
    return;
  end;
end;
if isempty(Beq) & ~isempty(Aeq)
  Beq=zeros(size(Aeq,1),1);
end;
nonlcon='';
if nargin>9 & ~isempty(nonlc)
  if ischar(nonlc)
    nonlcon=nonlc;
  else errmsg='fun must be a string.';
    return;
  end;
end;
settings = [0 0];
if nargin>10 & ~isempty(setts)
  if isnumeric(setts) & isreal(setts) & all(size(setts)<=size(settings))
    settings(setts~=0)=setts(setts~=0);
  else errmsg='settings should be a row vector of length 1 or 2.';
    return;
  end;
end;
maxSQPiter=1000;
%options = optimset('fmincon');
options = optimset(optimset('fmincon'),'MaxSQPIter',1000);
if nargin>11 & ~isempty(opts)
  if isstruct(opts)
    if isfield(opts,'MaxSQPIter')
      if isnumeric(opts.MaxSQPIter) & isreal(opts.MaxSQPIter) & ...
```

```
            all(size(opts.MaxSQPIter)==1) & opts.MaxSQPIter>0 & ...
            round(opts.MaxSQPIter)==opts.MaxSQPIter
          maxSQPiter=opts.MaxSQPIter;
          opts=rmfield(opts,'MaxSQPIter');
       else errmsg='options.maxSQPiter must be an integer >0.';
         return;
       end;
     end;
     options=optimset(options,opts);
  else errmsg='options must be a structure.';
     return;
   end;
end;

currentwarningstate=warning;
warning off;
tic;
lx = size(x0,1);
z_incumbent=inf;
x_incumbent=inf*ones(size(x0));
I = ceil(sum(log2(xub(find(xstatus==1))-xlb(find(xstatus==1))+1))+size(find(xstatus==1),1)+1);
stackx0=zeros(lx,I);
stackx0(:,1)=x0;
stackxlb=zeros(lx,I);
stackxlb(:,1)=xlb;
stackxub=zeros(lx,I);
stackxub(:,1)=xub;
stackdepth=zeros(1,I);
stackdepth(1,1)=1;
stacksize=1;
xchoice=zeros(size(x0));
if ~isempty(Aeq)
  j=0;
  for i=1:size(Aeq,1)
    if Beq(i)==1 & all(Aeq(i,:)==0 | Aeq(i,:)==1)
      J=find(Aeq(i,:)==1);
      if all(xstatus(J)~=0 & xchoice(J)==0 & xlb(J)==0 & xub(J)==1)
        if all(xstatus(J)~=2) | all(x0(J(find(xstatus(J)==2)))==0)
```

```
            j=j+1;
            xchoice(J)=j;
            if sum(x0(J))==0
              errmsg='x0 not correct.';
              return;
            end;
          end;
        end;
      end;
    end;
end;
errx=optimget(options,'TolX');
optionsdisplay=getfield(opts,'Display');
if strcmp(optionsdisplay,'iter') | strcmp(optionsdisplay,'final')
  show=1;
else
  show=0;
end;

while stacksize>0
  c=c+1;

  x0=stackx0(:,stacksize);
  xlb=stackxlb(:,stacksize);
  xub=stackxub(:,stacksize);
  x0(find(x0<xlb))=xlb(find(x0<xlb));
  x0(find(x0>xub))=xub(find(x0>xub));
  depth=stackdepth(1,stacksize);
  stacksize=stacksize-1;
  percdone=round(100*(1-sum(0.5.^(stackdepth(1:(stacksize+1))-1))));

  % user_update
  t=toc;
  if show
    disp(sprintf('    searched %3d %% of three',percdone));
    disp(sprintf('    z    : %12.4e',z_incumbent));
    %-------------------------------------------------------
    %zelf toegevoegd
```

```
  %disp(sprintf('    x    : %12.1d  %3.1d',x0(1), x0(2)));
  %-------------------------------------------------------
  disp(sprintf('    t    : %12.1f secs',t));
  disp(sprintf('    c    : %12d cycles',c-1));
  disp(sprintf('    fail : %12d cycles',fail));
end;

*** RELAXATION
[x z convflag output lambda]=fmincon(fun,x0,A,B,Aeq,Beq,xlb,xub,nonlcon,options,varargin{:});


*** FATHOMING

K = find(xstatus==1 & xlb~=xub);
separation=1;
%RCSP

if convflag<0 | (convflag==0 & settings(1))
  % FC 1
  separation=0;
  if show
    disp('    branch pruned');
  end;
  if convflag==0
    fail=fail+1;
    if show
      disp('    not convergent');
    end;
  elseif show
    disp('    not feasible');
  end;

elseif z>=z_incumbent & convflag>0
  % FC 2
  separation=0;
  if show
    disp('    branch pruned');
    disp('    ghosted');
```

```
    end;

 elseif all(abs(round(x(K))-x(K))<errx) & convflag>0
    % FC 3
    z_incumbent = z;
    x_incumbent = x;
    separation = 0;
    if show
       disp('    branch pruned');
       disp('    new best solution found');
    end;
 end;

% ***SELECTION

 if separation == 1 & ~isempty(K)
    dzsep=-1;
    for i=1:size(K,1)
       dxsepc = abs(round(x(K(i)))-x(K(i)));
       if dxsepc>=errx | convflag==0
          xsepc = x; xsepc(K(i))=round(x(K(i)));
          dzsepc = abs(feval(fun,xsepc,varargin{:})-z);
          if dzsepc>dzsep
             dzsep=dzsepc;
             ixsep=K(i);
          end;
       end;
    end;
    % *** SEPARATION
    if xchoice(ixsep)==0
        branch=1;
       domain=[xlb(ixsep) xub(ixsep)];
       sepdepth=depth;
       while branch==1
          xboundary=(domain(1)+domain(2))/2;
          if x(ixsep)<xboundary
             domainA=[domain(1) floor(xboundary)];
             domainB=[floor(xboundary+1) domain(2)];
```

```
    else
      domainA=[floor(xboundary+1) domain(2)];
      domainB=[domain(1) floor(xboundary)];
    end;
    sepdepth=sepdepth+1;
    stacksize=stacksize+1;
    stackx0(:,stacksize)=x;
    stackxlb(:,stacksize)=xlb;
    stackxlb(ixsep,stacksize)=domainB(1);
    stackxub(:,stacksize)=xub;
    stackxub(ixsep,stacksize)=domainB(2);
    stackdepth(1,stacksize)=sepdepth;
    if domainA(1)==domainA(2)
      stacksize=stacksize+1;
      stackx0(:,stacksize)=x;
      stackxlb(:,stacksize)=xlb;
      stackxlb(ixsep,stacksize)=domainA(1);
      stackxub(:,stacksize)=xub;
      stackxub(ixsep,stacksize)=domainA(2);
      stackdepth(1,stacksize)=sepdepth;
      branch=0;
    else
      domain=domainA;
      branch=1;
    end;
  end;
else
  % XCHOICE~=0
  L=find(xchoice==xchoice(ixsep));
  M=intersect(K,L);
  [dummy,N]=sort(x(M));
  part1=M(N(1:floor(size(N)/2))); part2=M(N(floor(size(N)/2)+1:size(N)));
  sepdepth=depth+1;
  stacksize=stacksize+1;
  stackx0(:,stacksize)=x;
  O = (1-sum(stackx0(part1,stacksize)))/size(part1,1);
  stackx0(part1,stacksize)=stackx0(part1,stacksize)+O;
  stackxlb(:,stacksize)=xlb;
```

```
        stackxub(:,stacksize)=xub;
        stackxub(part2,stacksize)=0;
        stackdepth(1,stacksize)=sepdepth;
        stacksize=stacksize+1;
        stackx0(:,stacksize)=x;
        O = (1-sum(stackx0(part2,stacksize)))/size(part2,1);
        stackx0(part2,stacksize)=stackx0(part2,stacksize)+O;
        stackxlb(:,stacksize)=xlb;
        stackxub(:,stacksize)=xub;
        stackxub(part1,stacksize)=0;
        stackdepth(1,stacksize)=sepdepth;
      end;
    elseif separation==1 & isempty(K)
      fail=fail+1;
      if show
        disp('    branch pruned');
        disp('    leaf not convergent');
      end;
    end;
end;

% OUTPUT
t=toc;
Z = z_incumbent;
X = x_incumbent;
disp(sprintf('\n    Branch and Bound completed'));
disp(sprintf('    time elapsed:              %12.1f secs',t));
disp(sprintf('    total cycles:             %12d cycles',c-1));
disp(sprintf('    cycles failed:             %12d cycles',fail));
disp(sprintf('    response value at optimum: %12.4e',z_incumbent));
disp(sprintf('    optimum design points for subproblem:\n'))
design = X(1:length(X)-1); disp([design]');
errmsg='';
%eval(['warning ',currentwarningstate]);
```

Bootstrap Related Codes

Code 1: Bootstrap_ninescrips.m--- this is the main code which calls the others functions

---

```
%profile on;
clear all;clc;
format short g;
[data, header] = xlsread('stocks_data_9scrips.xls');
[aa,bb] = size(data);
data = data(:,1:bb);
no_asset = bb;
kurse = data(:,:);
[n_K,N] = size(kurse);
LNkurse = log(kurse);
LNkurse_lag = lagmatrix(LNkurse, [1]);
rend = LNkurse(2:n_K,:) - LNkurse_lag(2:n_K,:);
rend = [rend;rend]; % for overlapping intervals
n_R = n_K - 1;
for i=1:no_asset
    figure
    plotdensity(rend(:,i),[0 0 0],[-0.1 0.1 0 40]);%plotting density of the returns, function call
end
print -depsc dr_1.eps;
bl = 20;
nob = ceil(n_R/bl);

BR = 500;% number of bootstraps
sizevech = 0.5*no_asset*(no_asset+1);
mu_sig_boot = zeros(BR,no_asset + sizevech);
tic

for b = 1:BR
  % build block bootstrap sample
  bfobs = ceil(rand(nob,1)*n_R);
  brend = zeros(bl*nob,no_asset);
  for nb = 1:nob
```

```
      brend((nb-1)*bl+1:nb*bl,:) = rend(bfobs(nb):bfobs(nb)+bl-1,:);
    end
    brend = brend(1:n_R,:);
    % calculate mean and covar
    mu_sig_boot(b,1:no_asset) = mean(brend);
    mu_sig_boot(b,no_asset+1:no_asset+sizevech) = vech(cov(brend));%Vech function call
    xyz(b,no_asset+1:no_asset+sizevech)=sqrt(mu_sig_boot(b,no_asset+1:no_asset+sizevech));
end
% here, we would have to calculate the mean and variance of
% the mu_sig_boot for the first few columns corresponding to
% the expected/mean values of the returns
toc
testi = [1 2 3 4 5 6 7 8 9 10 19 27 34 40 45 49 52 54];%first 9 are the estimated means of returns while the next 9 are the estimates of the variances.
for i=1:length(testi)
  figure
  plotdensity(sort(mu_sig_boot(:,testi(i))),[0 0 0]);%------>
  ausf = ['dmu_' int2str(testi(i))];
  print('-depsc',ausf);
  x = min(mu_sig_boot(:,testi(i))):(max(mu_sig_boot(:,testi(i)))-
  min(mu_sig_boot(:,testi(i))))/1000:max(mu_sig_boot(:,testi(i)));
  f = normcdf(x,mean(mu_sig_boot(:,testi(i))),std(mu_sig_boot(:,testi(i))));
  [h,p] = lillietest(mu_sig_boot(:,testi(i)),0.05)
  figure
  y = mean(mu_sig_boot(:,testi(i)))+randn(1000,1)*std(mu_sig_boot(:,testi(i)));
  qqplot(mu_sig_boot(:,testi(i)),y);
end
rejectionsmu = 0;
for i=1:9
    [h,p] = lillietest(mu_sig_boot(:,i),0.05);
    rejectionsmu = rejectionsmu + h;
end
rejectionsmu
rejections_var = 0;
for i=10:54
    [h,p] = lillietest(mu_sig_boot(:,i),0.05);
    rejections_var = rejections_var + h;
end
rejections_var
```

Code 2: Plotdensity.m, being called by the main code

```
function error = plotdensity(data,farbe,axisv);
obw = 0.9*1/(length(data)^0.2)*min(std(data),(data(round(length(data)*0.75))-data(round(length(data)*0.25)))/1.34);
if obw == 0
obw = 0.9*1/(length(data)^0.2)*std(data);
end
[f,x,u] = ksdensity(data,'kernel','box','width',obw);
plot(x,f,'-','LineWidth',1.5,'Color',farbe);
hold on;
pdf1 = normpdf(x,mean(data),std(data));
plot(x,pdf1,'--','LineWidth',1.5,'Color','g');
set(gca,'FontSize',10);
if nargin > 2
axis(axisv,'fill');
end
set(gca,'PlotBoxAspectRatio',[1 0.5 1]);
```

Code 3: vech.m, being called by the main code

```
function v = vech(x)
% PURPOSE: creates a column vector by stacking columns of x n and below the diagonal
%---------------------------------------------------------------------------
% USAGE:  v = vech(x)
% where:  x = an input matrix
% RETURNS:
%         v = output vector containing stacked columns of x
% Written by Mike Cliff, UNC Finance  mcliff@unc.edu, CREATED: 12/08/98
if (nargin ~= 1)
  error('Wrong # of arguments to vec');
end
[r,c] = size(x);
v = [];
for i = 1:c
  v = [v;x(i:r,i)];
end
```

GARCH Codes

GARCH1.m

```
% In this program the default model is GARCH(1,1) FOR 9 STOCKS
clear all;clc;
format short g;
[data, header] = xlsread('stocks_data.xls');
[aa,bb] = size(data);
for i=1:9
  stocki =  data(:,i);
  %plot daily price variation
  figure;
  plot([1:255],stocki)
  set(gca,'XTick',[10 50 90 130  170 210 245])
  set(gca,'XTickLabel',{'Feb 06' 'Apr 06' 'Jun 06' 'Aug 06' 'Oct 06' 'Dec 06' 'Feb 07' })
  ylabel('Price (EUR)')
  title('Price Variation')
 % Convert the prices to a return series
  stockireturnseries = price2ret(stocki);
  Mean(i,1) = nanmean(stockireturnseries);
  Median(i,1) = nanmedian(stockireturnseries);
  StandardDeviation(i,1) = nanstd(stockireturnseries);
  SampleVariance(i,1) = nanvar(stockireturnseries);
  Min(i,1) = nanmin(stockireturnseries);
  Max(i,1) = nanmax(stockireturnseries);
  Range(i,1) = range(stockireturnseries);
  k(i,1) = kurtosis(stockireturnseries);
  sk(i,1) = skewness(stockireturnseries);
  [H(i,1),P,JBSTAT(i,1),CV(i,1)] = jbtest(stockireturnseries);
  % Plot return series
  figure;
  plot([1:254],stockireturnseries)
  set(gca,'XTick',[10 50 90 130  170 210 245])
  set(gca,'XTickLabel',{'Feb 06' 'Apr 06' 'Jun 06' 'Aug 06' 'Oct 06' 'Dec 06' 'Feb 07' })
```

```
ylabel('Return')
title('Return Series')
%Check for correlation in the return series
figure;
autocorr(stockireturnseries)
title('ACF with Bounds for Raw Return Series')
%Check for correlation in the squared returns
figure;
autocorr(stockireturnseries.^2)
title('ACF of the Squared Returns')
%Ljung-Box-Pierce Q-Test for raw returns
[H_LM1(i,1), pValue_LM1(i,1), Qstat_LM1(i,1), CriticalValue_LM1(i,1)] = lbqtest(stockireturnseries-
mean(stockireturnseries),[15]',0.05);
%Ljung-Box-Pierce Q-Test for squared raw returns
[H_LM2(i,1), pValue_LM2(i,1), Qstat_LM2(i,1), CriticalValue_LM2(i,1)] = lbqtest((stockireturnseries-
mean(stockireturnseries)).^2,[15]',0.05);
% Engle's ARCH Test
[H_EA1(i,1), pValue_EA1(i,1), Qstat_EA1(i,1), CriticalValue_EA1(i,1)] = archtest(stockireturnseries-
 mean(stockireturnseries),[15]',0.05);
%%%%%%%%%%%%%%%%%%%%%%Parameter Estimation
[coeff,errors,LLF,innovations,sigmas,summary] = garchfit(stockireturnseries);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
garchdisp(coeff,errors)
%%%%%%%%% Postestimation Analysis
garchplot(innovations,sigmas,stockireturnseries)
figure;
plot(innovations./sigmas)
ylabel('Innovation')
title('Standardized Innovations')
figure;
autocorr((innovations./sigmas).^2)
title('ACF of the Squared Standardized Innovations')
H_LM3(i,1), pValue_LM3(i,1), Qstat_LM3(i,1), CriticalValue_LM3(i,1)] =
lbqtest((innovations./sigmas).^2,[15]',0.05);
end;
```

GARCH3.m

```
% In this code comparision has been made between the GARCH(P,Q)and% EGARCH(P,Q) with P& Q values can be changed for specific stocks so as to filter the autocorrelation of the Squared Standardized Innovations.
clear all;clc;
format short g;
[data, header] = xlsread('stocks_data.xls');
[aa,bb] = size(data);
for i=1:1
stocki =  data(:,i);
stockireturnseries = price2ret(stocki);
% Autocorrelation of the squared return series
figure;
autocorr(stockireturnseries.^2)
title('ACF of the Squared Returns')
% EGARCH(1,1)
spec = garchset('VarianceModel','EGARCH','M',1,'P',1,'Q',1);
spec = garchset(spec,'R',1,'Distribution','T')
[coeff,errors,LLF,innovations,sigmas,summary] = ...
   garchfit(spec,stockireturnseries);
   garchdisp(coeff,errors)
%GARCH (1,1)
%[coeff,errors,LLF,innovations,sigmas,summary] = ...
    % garchfit(stockireturnseries);
%GARCH(2,1)
 %spec21 = garchset('P',2,'Q',1);
 %[coeff,errors,LLF,innovations,sigmas,summary] = ...
     %garchfit(spec21,stockireturnseries);
% Autocorrelation of the Squared Standardized Innovations for the specific
% model [GARCH or EGARCH]
figure;
autocorr((innovations./sigmas).^2)
title('ACF of the Squared Standardized Innovations')

end;
```