Plan of Codes

Bi-Objective Reliability Based Design Optimization: Applications in Portfolio Investment Analysis

Raghu Nandan SENGUPTA[@]; Aditya GUPTA [#]; Subhankar MUKHERJEE[@]; Gregor WEISS[&]
[@]DoMS, Indian Institute of Technology Kanpur, INDIA, Email: raghus@iitk.ac.in
[#]McKinsey & Company, INDIA, Email: Aditya_Gupta-GUVA@mckinsey.com
[@]DoMS, Indian Institute of Technology Kanpur, INDIA, Email: subhankar@iitk.ac.in
[&]Faculty of Economics and Management, Leipzig University, GERMANY
Email: weiss@wifa.uni-leipzig.de

Note: Details of plan of codes, pseudo-codes and other set of detailed runs results (not discussed in this paper) are all give in the open access link, <**https://github.com/RNSengupta/Bi-Objective_RBDO_Paper**> which any interested reader can refer

Multi-Objective Optimization Code (Where Relevant Changes for Model II, and III may be Incorporated as needed)

Process of running

01) Go to the directory of model we want to run. Example if we want to run Mean- Variance model with reliability on return constraint go to the directory Codes/Multiobjective/Classical Markowitz with Resampling/Model I(A)/1P/Return.

02) Now each model directory have a script with its model name as a title. Run that script to start the optimization. E.g.,. Run the **RBDO with Return** script in python to run the code.

03) Each directory has **Bootstrap_data** ,**Log – Return** and **Covariance** excel file which consist of respective results.

Description of main code

01) First import the libraries. If libraries are not installed (an error message will popup in that case) please install it.

02) For the optimization model all the libraries functions are required and must be included.

03) Code reads from the excel file **Bootstrap_data.xls** the data for the variance covariance matrix and the means of the all the N=30 stocks.

04) Each sub-directories specified will have another python script required for multiobjective optimization algorithm.

05) The code **Classical** is called which contains objective function and constraints. This will be an input to optimizing function for minimizing or maximizing.

06) The algorithm vector consist of all the parameters required for optimization.

Multiobjective Optimization

01) The optimization process is divided into two scripts first is main script in which optimization parameters are defined and we call the optimization problem, and in second script the optimization problem for different risk metrics is defined, e.g., **RBDO with Return** is the first script and **Classical.py** is the second script.

02) **Classical.py** defines the deterministic multiobjective model. The problem can be written as Classical(ret,cov,0,1) i.e., ret is return vector, cov is covariance matrix, 0 is the r_p* i.e. the lower bound of portfolio return and $\sigma^*_{i,j\ P}$ is the upper bound of the portfolio risk.

03) The outputs are: (i) two objective function values, (ii) weight matrix. The sum of weights in output is not equal to 1 because we randomly take all weight variable values between 0 and 1 and save this as the input given to optimization. But in analysis part we divide the weight vector by its sum. Hence when output it returned, it returns original vector, so it should be divided by its sum (normalization step as required).

Reliability based Optimization

In RBDO there are two loops. The first loop is used for the multi-objective problem solution considering deterministic input while the second loop is the PMA approach which find the MPP for the decision variables. Together the form the SORA method

01) After one assigns a fixed reliability index value (β) we assign initial inputs for reliability part which equals deterministic results of the multi-objective solition.

02) For inner loop there are two conditions to be satisfied, **e_val** i.e., difference between two consecutive optimization steps (which should be greater than ε (an assigned value)), and the maximum number of iteration which is t be followed.

03) We incorporate uncertainty by **Reliable_ret** and **Reliable_sd** function for mean and standard deviation.

04) We again simulate the multiobjective optimization problem with new set of parameters and **pymo_opt.minimize** function is used to optimize the problem.

05) One finds the optimal portfolio by the chosen metric e.g., by considering the maximum of the Sharpe ratio or Return/Risk ratio, depending on the Model we are solving.

06) After the optimization is completed the final result are stored in **result_f** variable.

07) Once the optimization is completed we plot the Risk, Return, beta, weight graphs as required.

Description about Reliable_sd function

For calculating the robust return the following function is used

01) This function calls **Rosenblatt** function for Rosenblatt transform.

02) Than it optimizes mpp objective function **get_mpp_objective_sd** with **mpp_constriant** function which are formed according to PMA method.

03) Optimization is performed by **sci_opt.minimize** function.

04) The result obtained transformed with the help of **Inverse_Rosenblatt** function and used as an input for Reliability based optimization.

Plan of Codes

Bi-Objective Reliability Based Design Optimization: Applications in Portfolio Investment Analysis

Raghu Nandan SENGUPTA[@]; Aditya GUPTA [#]; Subhankar MUKHERJEE[@]; Gregor WEISS[&]
[@]DoMS, Indian Institute of Technology Kanpur, INDIA, Email: raghus@iitk.ac.in
[#]McKinsey & Company, INDIA, Email: Aditya_Gupta-GUVA@mckinsey.com
[@]DoMS, Indian Institute of Technology Kanpur, INDIA, Email: subhankar@iitk.ac.in
[&]Faculty of Economics and Management, Leipzig University, GERMANY
Email: weiss@wifa.uni-leipzig.de

05) **Rosenblatt Transform** can be done either by defined function **Rosenblatt** or predefined function **cp.MvNormal()** , Both gives the same results and same thing for **Inverse Rosenblatt.**

Bootstrap Related Code

For the Bootstrap related codes go to the directory Codes/Bootstrap and open Bootstrap.py

01) **Bootstrap** script reads the **Log – Return** file and has **Analyse** function which are all used to find the mean ,variance , standard deviation and skewness.

02) After that find the optimal block length for block bootstrap and the value for the same is saved as **b_star_cb**.

03) **CicularBlockBootstrap** function is used to create bootstrap mean, variance , standard deviation and skewness values and the data is saved in **Bootstrap_data.xlsx.**

04) One then utilizes the data to create the covariance matrix which is saved as **Covariance.xlsx**

05) The following functions are used to calculate **Skew Relation Matrix** and **Coskewness** Matrix.

Testing the Initial Data

01) In this code first end of the day stock price is concerted into **Log-return.**

02) After that **ADF Test , ARCH Multiplier Test, Ljung-Box, Jarq – Bera Test** are preformed and **Kernal Density plot** , **ACF** and **PACF** plots are plotted and saved in Plots folder.