Paper 3225-2015

Portfolio Construction with OPTMODEL

Robert Spatz, University of Chicago; Taras Zlupko, University of Chicago

ABSTRACT

Investment portfolios and investable indexes determine their holdings according to stated mandate and methodology. Part of that process involves compliance with certain allocation constraints. These constraints are developed internally by portfolio managers and index providers, imposed externally by regulations, or both. An example of the latter is the U.S. Internal Revenue Code (25/50) concentration constraint, which relates to a regulated investment company (RIC). These codes state that at the end of each quarter of a RIC's tax year, the following constraints should be met: 1) No more than 25 percent of the value of the RIC's assets might be invested in a single issuer. 2) The sum of the weights of all issuers representing more than 5 percent of the total assets should not exceed 50 percent of the fund's total assets. While these constraints result in a non-continuous model, compliance with concentration constraints can be formalized by reformulating the model as a series of continuous non-linear optimization problems solved using PROC OPTMODEL. The model and solution are presented in this paper. The approach discussed has been used in constructing investable equity indexes.

INTRODUCTION

A multitude of methodologies can be presented defining ways of pooling assets into single investment portfolio. While there are countless ways to build portfolios, there are some common requirements that have to be met and are imposed on regulated investment companies representing these managed portfolios of assets. U.S. Internal Revenue Code stipulates the idea of portfolio diversification by imposing a set of requirements on regulated investment company which can be formulated as follows:

1. No more than 25 percent of the value of portfolio assets might be invested in a single issuer.

2. The sum of the weights of all issuers representing more than 5 percent of the total assets should not exceed 50 percent of the total portfolio assets.

The above constraints should be met quarterly and are relevant to such investment vehicles as mutual funds, exchange traded funds and others as well as indexes designed to serve as benchmarks of such investment entities.

There are different ways to accommodate the regulation-imposed concentration constraints described above in investment portfolio. Some of these are discussed below.

THE MODEL

Once portfolio construction methodology is defined, the regulation-imposed diversification requirements can be accommodated in a variety ways. Having more than twenty components in a portfolio, equal-weighting the portfolio will satisfy the concentration constraints. However, this will likely change the nature of portfolio substantially unless it is a capitalization-weighted portfolio where the portfolio components have similar market capitalization.

A way to remain close to originally defined portfolio is to start with securities violating the concentration constraints. For example, reduce the weight of securities which constitute more that 25% of portfolio weight and redistribute the excess weight proportionately to remaining securities. Then proportionately reduce the weights of those securities counted towards the 50% constraint and redistribute to smaller-weighted securities.

The idea of least departure from the original portfolio composition while complying with regulation-stipulated diversification requirements can be formalized in the following non-linear mathematical optimization model:

$$min \sum_{i=1}^{N} (w_i - x_i)^2 \quad (1)$$

subject to

$$\sum_{i=1}^{N} x_i = 1 \quad (2)$$

$$0 \leq x_i \leq 0.25 \quad (3)$$

$$\sum_{i:\, x_i > 0.05} x_i \leq 0.5 \quad (4)$$

where

w_i' – weight of i-th security in original portfolio

x_i – weight of i-th security in regulation-compliant portfolio

N – number of securities in portfolio

Conditions (3) and (4) above specify the compliance to the 25/50 concentration requirements. Equation (2) sets the sum of all weights to unity reflecting the fact that total of all portfolio components should add-up to 100%. The objective function (1) minimizes the square of differences between the initially-defined non-compliant portfolio and the one which will comply with the regulation-imposed diversification requirements.

The model above can be solved using the SAS® OPTMODEL procedure which minimizes objective function (1) given the constraints (2)-(3). However, constraint (4) on the sum of securities greater than five percent introduces non-continuity leading to inconsistent results between different runs of PROC OPTMODEL. This is because non-linear solvers are designed to handle only smooth problems where objective and constraint functions are continuous and differentiable in order for the optimization algorithm to consistently converge.

The issue of non-continuity of the model (1)-(4) can be tackled by reformulating the process such that PROC OPTMODEL is called ten times and then choosing the result with the lowest objective value. The ten calls are made with zero to nine largest securities and the remainder smallest securities. The choice of ten calls (nine “large” securities as the maximum) is because of the ratio between 5% and 50%; there can never be more than nine securities whose weights exceed 5% where their summed weights are less than or equal to 50%. Generic and practical solutions of the model above are discussed in the chapters that follow.

PROC OPTMODEL SOLUTION

PROC OPTMODEL provides a powerful and versatile modeling environment for building and solving mathematical optimization models. It was introduced with SAS/OR version 9.1.3 and now includes solvers for linear, mixed integer, quadratic and general non-linear programming problems. The programming language of PROC OPTMODEL is intuitive and transparent closely mimicking the symbolic algebra of optimization model formulation.

The code below shows the generic implementation of the earlier defined portfolio optimization model in PROC OPTMODEL. Please note, that constraint (3) of the model is accommodated in the PROC OPTMODEL code by setting bounds to the weights at the variable declaration step.

Initial non-regulatory compliant weights of portfolio components in the code below are located in data set 'weights'. There are fifty securities in the portfolio being optimized.

The code begins from invoking PROC OPTMODEL and proceeds with reading the set of portfolio weights, defining objective and constraints and creating dataset of initial and final security weights as follows:

```
/*invoke procedure*/
proc optmodel;

/*define count of securities*/
%let count=50;

/*read in weights of securities into an array w*/
set<number> nn;
number w{1..&count};
read data weights into nn=[_n_] w;

/*declare optimization variable and set bounds*/
var x{i in 1..&count} >=0 <=0.25;

/*declare minimization objective*/
min   z=sum{i in 1..&count}((w[i]-x[i])^2);

/*declare constraints*/
constraint c1:sum{i in 1..&count} (if x[i]>0.05 then x[i] else 0) <=0.5;
constraint c2:sum{i in 1..&count} x[i]=1.0;

/*invoke the NLPC nonlinear programming solver*/
solve with nlpc / tech=quanew;

/*create dataset of initial and final weights*/
create data results from [_n_] w x;

quit;
```

While the code above provides an illustration of implementing all the IRS diversification constraints in PROC OPTMODEL, as currently formulated and depending on the input data, may result in non-convergence of results in some cases.

PORTFOLIO OPTIMIZATION EXAMPLE

The model and the code presented in the earlier sections provide a way to approach the solution of the 25/50 portfolio diversification requirement. This section includes the code used to conduct practical optimization of a hypothetical portfolio which is by design not compliant to the 25/50 diversification requirements. The portfolio includes less than one hundred stocks, spans over ten years and is reconstituted on the last trading day of each calendar quarter.

While model (1)-(4) and the subsequent code represent the goal of optimization and constraints well, constraint (4) of the model introduces discontinuity. This makes the optimization algorithm converging to inconsistent final optimal set of values each time the optimization is performed. A way to resolve this is to implement a process that takes into account the nature of this constraint.

Constraint (4) of the model states that the sum of weights of securities larger than five percent of the total portfolio assets should not exceed fifty percent of total portfolio assets. Therefore, there could be maximum nine securities with weights larger than five percent when portfolio is still compliant to constraint (4). In this context large portfolio components weighting up to 25% of total portfolio and small ones weighting up to 5% are treated separately in the model both in objective function and in constraints. The code below shows the implementation of this approach as SAS macro.

The following is performed:

1. The dataset of initial weights is split up into two: One that includes large weights and another that includes small weights
2. Invoke/run PROC OPTMODEL
3. The results of PROC OPTMODEL are merged into a single dataset
4. Relevant data for each of the ten runs are added into a single dataset

An optimization run producing solution based on one application of PROC OPTMODEL is included in the following macro:

```
%macro run_one(inp, n);
/* inp - input data set, must be sorted in descending weight
   n  -  the number of "large" companies - a number from 0 to 9 */

/* 1. split the input dataset into large and all other securities*/

data tmp.in_large_&n. (keep=permnol iwl) tmp.in_small_&n. (keep=permnos iws);
set &inp.;
length permnol iwl permnos iws 8;
if (_N_ <= &n.) then do;
      permnol     = permno;
      iwl         = iw;
      output      tmp.in_large_&n.;
end;
else do;
      permnos     = permno;
      iws         = iw;
      output      tmp.in_small_&n.;
end;
run;

/* 2. run proc optmodel */

proc optmodel printlevel=2; /* printlevel=2 outputs all tables */
ods output SolutionSummary=tmp.sol_&n. ProblemSummary=tmp.prob_&n.
OptStatistics=tmp.opt_&n.;  /* have the output go to temp tables */

number ub_one;  /* upper bound for a single company (25%) */
number topc;    /* breakpoint for large versus small (5%) */
number ub_topc; /*upper bound sum of the weights for all large companies
(50%) */
number lb_one;  /* The minimum weight for a single company (0%) */
number tot;    /* The sum of the weights of all the companies (100%)  */

/* avoid code changes by reading parameters from table */
read data inp.params into ub_one topc ub_topc lb_one tot;

/* read in the companies in the large dataset into an array iwl (initial
weights large) */
set<number>permnol;
number iwl{permnol};
read data tmp.in_large_&n. into permnol=[permnol] iwl;

/* read in the companies in the small dataset into an array iws (initial
weights small) */
set<number>permnos;
number iws{permnos};
read data tmp.in_small_&n. into permnos=[permnos] iws;
```

```
/* define the constraints of the model */
/* cwl (constrained weights large) have to be between 0% and 25% */
var cwl{permnol} >= lb_one <= ub_one;
/* cws (constrained weights small) have to be between 0% and 5% */
var cws{permnos} >= lb_one <= topc;
/* the sum of all large have to be less then max sum (50%) */
constraint  con_topc: sum{i in permnol}   cwl[i] <= ub_topc;
/* all companies have to add up to 100% */
constraint  con_tot:  sum{i in permnol}cwl[i] + sum{j in permnos}cws[j]=tot;

/* the optimization function - minimize the squared difference between the
input weight and the constrained weight */
min z = sum{i in permnol}((iwl[i] - cwl[i])**2) + sum{j in permnos}((iws[j] -
cws[j])**2);
/* invoke the NLPC solver */
solve with nlpc;

/* output the large companies to a dataset */
create data tmp.out_large_&n. from [permnol]={i in permnol} iwl cwl;
/* output the small companies to a dataset */
create data tmp.out_small_&n. from [permnos]={j in permnos} iws cws;
quit;

/* merge the large company and small company datasets together*/
data tmp.data_&n.;
set tmp.out_large_&n. (rename=(permnol=permno iwl=iw cwl=cw))
    tmp.out_small_&n. (rename=(permnos=permno iws=iw cws=cw));
length      err nlarge 8;
err         = (iw - cw)**2;
nlarge      = &n.;
run;

/* 3. merge the needed details of the optmodel into a single observation*/

data tmp.stat_&n. (keep=nlarge solvtime numvar objval iterations solstat
solver);
set tmp.prob_&n. (drop=cValue1) tmp.sol_&n. tmp.opt_&n.(drop=cValue1);
      length nlarge solvtime numvar objval iterations 8;
      length      solstat $10 solver $20;
      retain solvtime numvar objval iterations solstat solver;
      format objval 11.8;

      nlarge      = &n.;
      if (Label1 = 'Number of Variables')
      then numvar = nValue1;
      else if (Label1 = 'Objective Value')
            then objval = nValue1;
      else if (Label1 = 'Iterations')
            then iterations   = nValue1;
      else if (substr(Label1,1,19) = 'Absolute Optimality')
            then absopterr = nValue1;
      else if (substr(Label1,1,19) = 'Relative Optimality')
            then relopterr = nValue1;
      else if (Label1 = 'Absolute Infeasibility')
            then absinf = nValue1;
      else if (Label1 = 'Relative Infeasibility')
```

```
            then relinf = nValue1;
      else if (Label1 = 'Solution Status')
            then solstat = cValue1;
      else if (Label1 = 'Solver')
            then solver = cValue1;
      else if (Label1 = 'Solver Time') then do;

                  solvtime    = nValue1;

                  output;
      end;
run;

/* 4. append the data from all ten runs in a single dataset use later */

proc append base=tmp.data_all data=tmp.data_&n.;
proc append base=tmp.stat_all data=tmp.stat_&n.;
run;
%mend run_one;
```

The macro above is run ten times where the number of large companies parameter n is changed from zero to nine. This is to reflect the fact that portfolio may include anywhere from zero to nine securities allowed to be large enough to count as contributing to the fifty percent constraint. Optimal compliant portfolio composition is chosen based on best value of objective function coming out of these ten runs.

For the purpose of comparisons, in addition to the optimized portfolio a version of diversification-compliant portfolio was also constructed by simply decreasing the weights of any security above 25% to the maximum compliant level and then proportionally distributing weights among smaller securities. Chart below shows rolling correlations between the optimized and simple proportional weight and the original non-compliant portfolio. As evident in Figure 1, out of the two regulation-compliant portfolios the optimized portfolio on average shows higher correlation with the original non-compliant portfolio then diversified portfolio adjusted by simple proportional method.

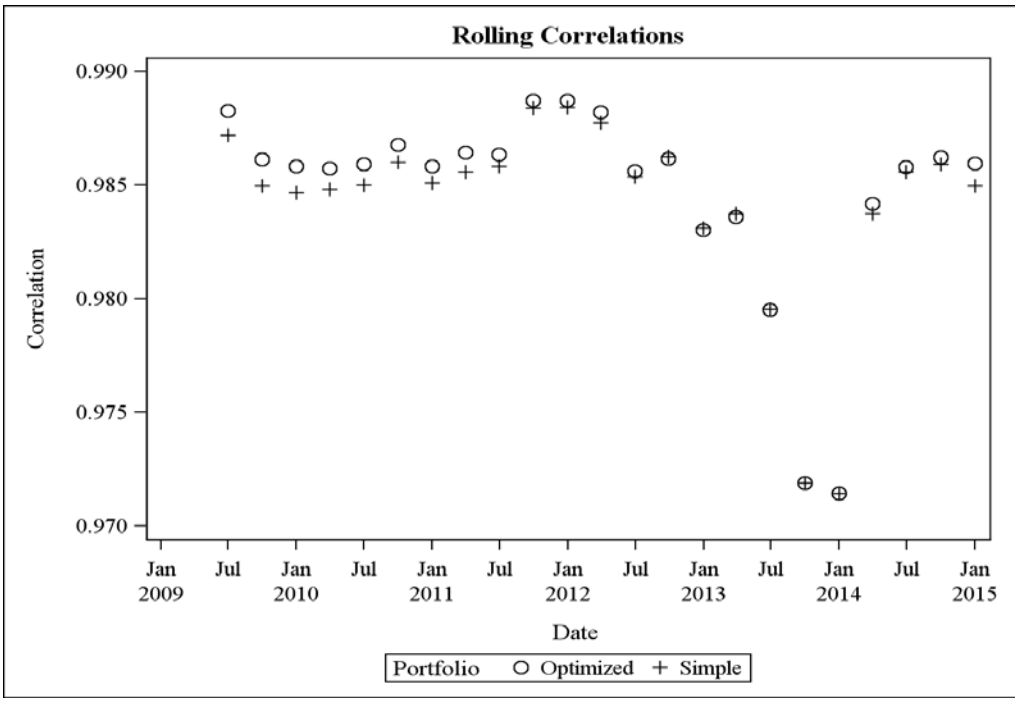

Figure 1. Rolling Correlations Between Initial and Optimized and Simple Portfolios

CONCLUSION

This paper provides an illustration of practical application of SAS PROC OPTMODEL when a non-linear mathematical optimization model built to reflect equity portfolio compliance with the regulation-imposed diversification requirements needs to be solved. The approach to address the problem, the model and the PROC OPTMODEL language is discussed. A way to address the issue of model non-continuity while solving the model with PROC OPTMODEL is also presented. An example of optimizing hypothetical portfolio is presented and shows advantages of using optimization. The modeling and solution approach using PROC OPTMODEL discussed in the paper has been used in building investable equity indexes.

RECOMMENDED READING

- *SAS/OR® 13.2 User's Guide, Mathematical Programming*
- *IRS Code Title 26 section 851(b)(3) http://www.gpo.gov/fdsys/pkg/USCODE-2010-title26/pdf/USCODE-2010-title26-subtitleA-chap1-subchapM.pdf*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Taras Zlupko
University of Chicago
taras.zlupko@crsp.ChicagoBooth.edu
http://crsp.chicagobooth.edu/