A Graduated Smoothing Method for the Portfolio Allocation Problem with Transaction Costs

by

Su Yan

A research paper
presented to the University of Waterloo
in partial fulfilment of the
requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Prof. Thomas F. Coleman

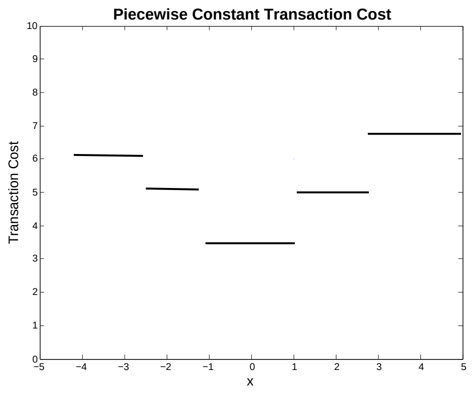
Waterloo, Ontario, Canada, 2016

Abstract

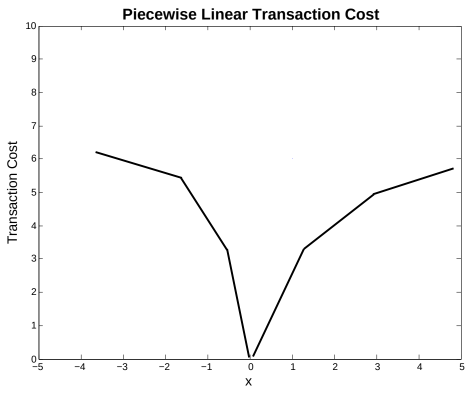
In this paper, our goal is to solve the piecewise-constant transaction cost minimization problem, one of the practical problems in finance, and to illustrate its computational difficulty. We develop and apply a smoothing technique for this problem. Finally, we compare examples of various dimensions to demonstrate the potential of this new smoothing method for this global optimization problem.

1. Introduction

In stock markets, transaction costs are fees that arise when selling or purchasing assets, and they are often piecewise-constant or piecewise-linear, illustrated in the following figures, where x is an asset holdings.



(a) Piecewise Constant Function



(b) Piecewise linear Function

The piecewise constant transaction cost implies the transaction costs stay the same within an interval. And the piecewise linear transaction cost means the transaction costs are proportional to the purchasing amount within an interval, and the proportion is decreasing as the purchasing amount increases.

In this paper, we deal with the case when the transaction costs are piecewise-constant, a common situation. The piecewise constant functions are discontinuous, which are hard to minimize and in particular, smooth minimizing processes do not apply. Hence, we modify the transaction cost in two stages via linear interpolations and quadratic splines to get a continuous and differentiable approximation. In the first stage, we use linear interpolation to fill the "gap" between levels and make it continuous, which is denoted by the function $TC_\Delta(x)$.

For $x \geq 0$,

$$TC_\Delta(x) = \begin{cases} level1 + \frac{\delta*(x-k+\Delta)}{2*\Delta} & \text{if } k-\Delta \leq x \leq k+\Delta \text{ and } \Delta \neq 0 \\ level1 & \text{if } k-\Delta \leq x \leq k+\Delta \text{ and } \Delta = 0 \\ level1 & \text{if } x < k-\Delta \\ level2 & \text{otherwise,} \end{cases}$$

where $level1$ and $level2$ represents the value of the first level and the second level respectively, and k is the "jump" point from $level1$ to $level2$, and $\delta = \frac{(k-\Delta)}{k} * (level2 - level1)$.

For $x < 0$, $TC_\Delta(x) = TC_\Delta(|x|)$.

In the second stage, we use quadratic splines to smooth the corners and make the function differentiable. The smoothed function is denoted by the function $TCb_\Delta(x)$.

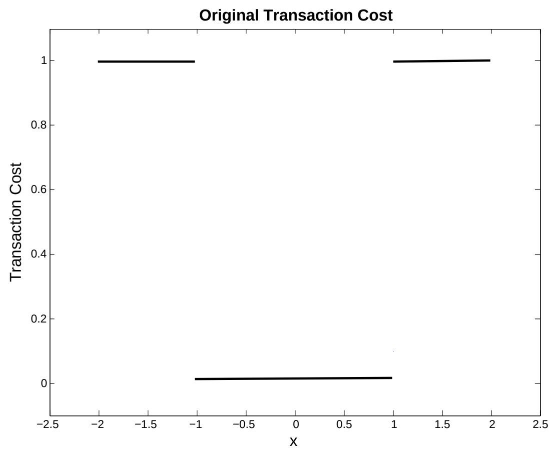
For $x \geq 0$,

$$TCb_\Delta(x) = \begin{cases} level1 + \frac{m}{4*\epsilon*(x-(k-\Delta)+\epsilon)^2} & \text{if } k-\Delta-\epsilon \leq x \text{ and } x \leq k-\Delta+\epsilon \text{ and } \Delta \neq 0 \\ level1 & \text{if if } k-\Delta-\epsilon \leq x \leq k-\Delta+\epsilon \text{ and } \Delta = 0 \\ level1 + \delta - \frac{m}{4*\epsilon*((k+\Delta)-x+\epsilon)^2} & \text{if } k+\Delta-\epsilon \geq x \text{ and } x \leq k+\Delta+\epsilon \text{ and } \Delta \neq 0 \\ level1 & \text{if } k+\Delta-\epsilon \geq x \text{ and } x \leq k+\Delta+\epsilon \text{ and } \Delta = 0 \\ level1 & \text{if } x < k-\Delta-\epsilon \\ level2 & \text{if } x > k+\Delta+\epsilon \\ TC_\Delta(x) & \text{otherwise,} \end{cases}$$

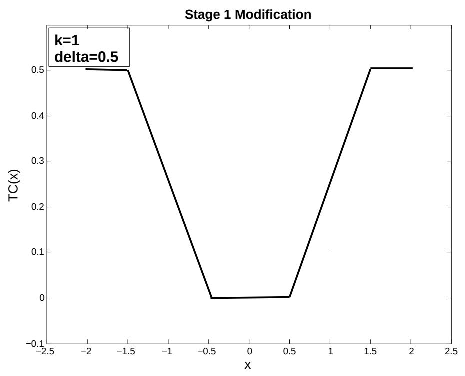
where $\epsilon = \frac{\Delta}{10}$, and $m = \frac{\delta}{2*\Delta}$.

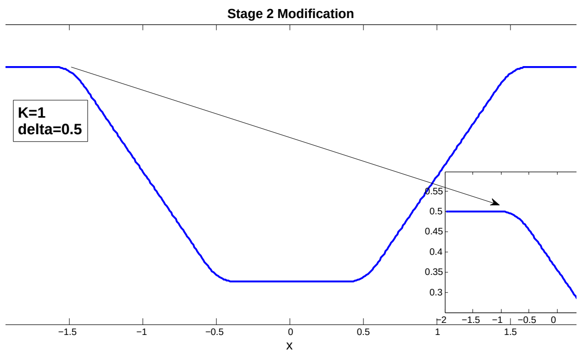
For $x < 0$, $TCb_\Delta(x) = TCb_\Delta(|x|)$.

The following figures illustrate this modification process.



(a) Original Transaction Cost





(b) TC(x)

(c) TCb(x)

Figure 2: Transaction Modification Process

In finance, the usual portfolio allocation problem is of the form

$$min\{1/2 * x' * H * x : Ax = b, x \geq 0\}, \tag{1}$$

where the term $1/2 * x' * H * x$ represents risk, and the expected return of the portfolio is expressed in one of the constraints. Thus, the problem is to minimize risk under certain constraints. Since the risk function is smooth and convex, we can solve it by quadratic programming.

In this paper, we would like to minimize the risk as well as the transaction costs. Hence, we add a term $S(x)$ representing the summation of the transaction costs into the objective function of equation (1), which gives us the following problem.

$$min\{S(x) + 1/2 * x' * H * x : Ax = b, x \geq 0\} \tag{2}$$

However, $S(x)$ and $1/2 * x' * H * x$ have different units, and hence, it is problematic to add these two terms. One way to adjust the unit difference is by weighting the two terms differently, which gives the following problem

$$min\{w * S(x) + 1/2 * x' * H * x : Ax = b, x \geq 0\}, \tag{3}$$

where w is the weight.

Another way to adjust it is replacing $1/2 * x' * H * x$ by the term $\sqrt{1/2 * x' * H * x}$ and gives the following minimization problem. We will focus on this formulation in the paper.

$$min\{S(x) + \sqrt{1/2 * x' * H * x} : Ax = b, x \geq 0\} \tag{4}$$

In the financial markets, we have the following interpretation. We hold a set of n assets, which is represented by a $n \times 1$ vector x indicating the amount we hold for each asset and is assumed to be nonnegative. The problem (4) we are facing is to minimize the total cost of the transaction costs and the risks under certain constraints. In this paper, we use the $S_{\Delta}(x) = \sum TCb_{\Delta}(x)$, the summation of the modified transaction costs to approximate $S(x)$. And the risk is demonstrated by the later term, namely $\sqrt{1/2 * x' * H * x}$, where H is a positive definite $n \times n$ matrix and is known as the covariance matrix. The constraint $Ax = b$ represents certain restrictions on x, where A is an $m \times n$ matrix and b is an $m \times 1$ vector. Matrix A has at least two rows. One row represents the total wealth constraint, that is, how much money is to be invested. Hence, each element in this row equals unity and the corresponding right-hand-side element (i.e., corresponding component of vector b) is the available sum to be invested. Another required row represents the target return of the portfolio, that is, the expected return of portfolio 'x'. In this row, the coefficients represent the expected return of the corresponding asset, which are estimated using historical data or scenario analysis, and the corresponding right-hand-side element (i.e., the corresponding element of b) is the user-chosen desired expected return of the portfolio (or target value of the portfolio). Other rows of matrix A represent additional constraints on groups of variables to restrict investments in certain sectors.

For a given $\Delta > 0$, the optimization problem we solve is:

$$min\{\sum S_{\Delta}(x) + \sqrt{1/2 * x' * H * x} : Ax = b, x \geq 0\} \tag{5}$$

For example, $x = [1; 2; 5; 0]$ means that we have 1 unit of the first asset and 2 units of the second asset and 5 units of the third asset and do not have any asset 4. If we have the constraint $A = [1, 1, 1, 1; 1.09, 0.98, 0.92, 1.1]$ and $b = [1; 1.2]$, it means that we use one unit of money to buy the four assets, and the four assets have expected rates of return 1.09, 0.98, 0.92 and 1.1, respectively, and our expected return of the portfolio is 1.2.

Problem (4) can be solved by the smooth minimizing methods. However, problem (4) is not convex, therefore standard local minimization methods may find the nearest local minimizer instead of the actual global one. Thus, in this paper, we introduce a smoothing technique to try to overcome this difficulty. We will show that the new method can often obtain better results than the standard local methods for most of the cases in the later section.

2. Transaction Cost Function Minimization Problem

In this section, we will first provide a description of the transaction cost function minimization problem and then introduce its difficulties.

2.1. Description of the Problem

In this paper, we solve the problems of the form $min\{S(x) + \sqrt{1/2 * x' * H * x} : Ax = b, x \geq 0\}$, where $S(x)$ is the summation of the transaction costs, $S(x)$ is approximated by $S_\Delta(x)$, the summation of the modified transaction cost function that is piecewise-linear with quadratic splines, and H is a positive definite matrix. For simplicity, we only consider the piecewise constant transaction cost of two levels with the lower level to be zero and the upper level to be one, which is demonstrated in the following figure and will result in relatively hard problem.

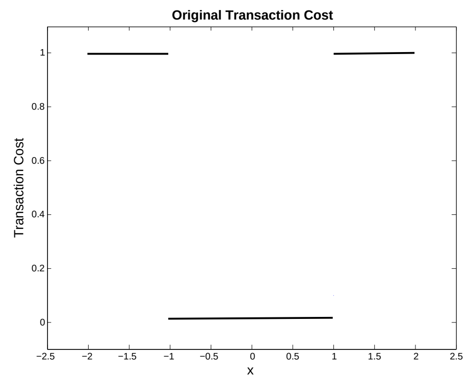


Figure 3: Simplified Transaction Cost Model

We have two types of constraints on x in the problem. The inequality constraint

$x \geq 0$ is to prevent negative entries in x, which means that we can only hold non-negative amount of assets. It is reasonable to only allow holding nonnegative amount of assets in the stock markets or other financial markets when 'shorting' is not allowed. We also have a set of equality constraints, which is represented by the equation $Ax = b$, where A is a matrix and b is a vector. The first row of A and b represents the wealth constraint, where the entries are all 1's meaning we use one unit of wealth to buy the set of asset, i.e. $e' * x = total\ wealth$. The second constraint is the target expected return of the portfolio, that is, $w' * x = target\ expected\ return$, where $w(i)$ is the expected return of asset i determined empirically, typically using historical data. Other constraints may represent other conditions, such as sector constraints, which indicate the proportion we invest in certain sector. For example, 25% of the investment is allocated to the technology assets.

2.2. Difficulties of the Problem

The minimization problem (5) can be solved by applying smooth local minimizing methods. But such methods will lead, typically, to the nearest local minimizer. In order to avoid this, and attempt to reach a global minimizer, we solve a sequence of problems (3), with decreasing values of Δ. Each new minimization (i.e. new value of Δ) uses as a starting point to minimizer of the previous problem. In this subsection, we will discuss the difficulties of the problem and actually make it visualize to the reader by plotting 2 dimension examples.

Our problem includes two parts. The first part is the summation of the transaction costs and the second part is the risk part. We illustrate these two parts separately using the following example.

We let $A = [1, 1, 1, 1; 1.3, 1.1, 0.98, 0.99]$ and $b = [1; 1.2]$, and with the equality constraint $Ax = b$, we solve for x. Thus, in order to satisfy the constraint, x must equal to $[0.5 + 0.6 * z + 0.55 * w; 0.5 - 1.6 * z - 1.55 * w; z; w]$, for some z and w. Then, we plot the $S(z, w)$ with respect to z and w, which gives the following figure.

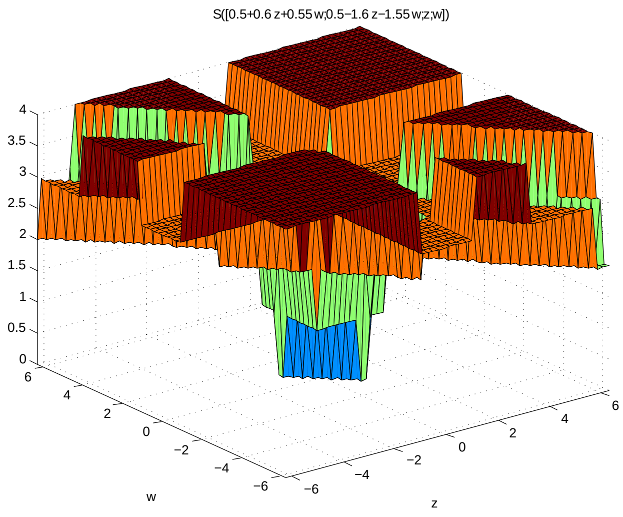


Figure 4: S function

And we randomly generate a positive definite matrix H of dimension 4×4, which gives the matrix

$H = [0.4389,\ -0.1161,\ -0.3817,\ 0.0284;$

$-0.1161,\ 0.6417,\ 0.2276,\ -0.1544;$

$-0.3817,\ 0.2276,\ 0.4457,\ 0.2048;$

$0.0284,\ -0.1544,\ 0.2048,\ 0.8113];,$

and we plot the risk term, namely $\sqrt{1/2 * x' * H * x}$, where x is of the form $[0.5 + 0.6 * z + 0.55 * w; 0.5 - 1.6 * z - 1.55 * w; z; w]$, for some z and w, which gives the following figure.

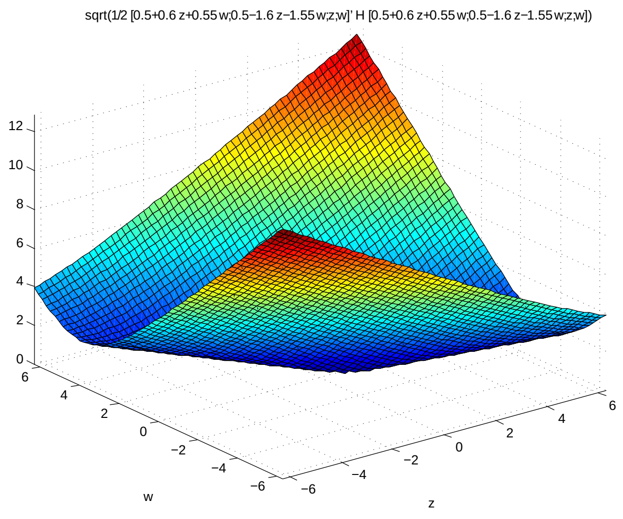


Figure 5: Risk function

And then, we combine those two parts together and plot the function that we are trying to minimize.

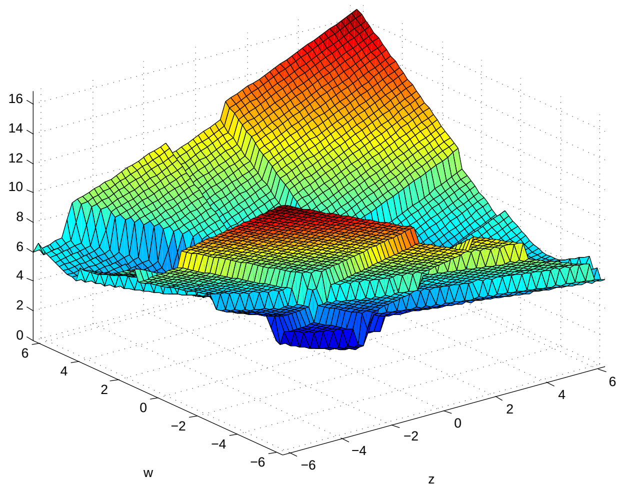


Figure 6: Combination

We can see from the example that although the risk part is a smooth function, the problem still has a challenging structure. The reason is although we modified the transaction cost function to make it continuous and differentiable, it is not convex, which makes global minimization hard, since it is not guaranteed to have a global minimizer. From the above figure, we can easily see that there are many local minimizers. If starting from a random point, the smooth minimization method are more likely to find some nearby local minima or perhaps diverge. Hence, the transaction cost part makes the problem hard to solve.

3. Smoothing Minimization Procedure

In the previous section, we see that the problem, even in low dimension, is hard to solve by smooth minimizing procedures. Thus, in this section, we will describe the smoothing minimization procedure, which will overcome the difficulties in most of the cases. Previous related work can be found in the papers [1], [2].

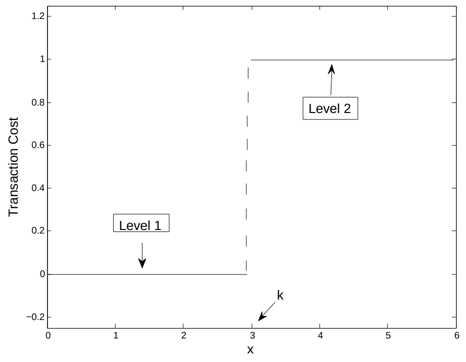
We first show the algorithm.

Algorithm 1 Minimization Procedure with Smoothing Technique

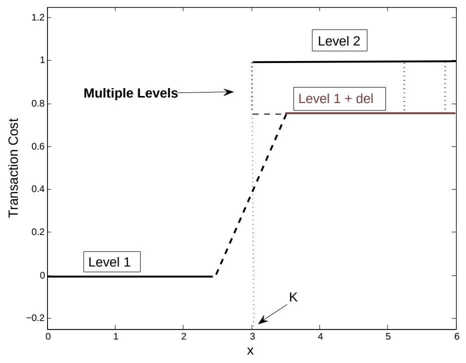
```
1: Find the global minimizer when $\Delta = 1$ by quadratic programming, denoted by $z$ ▷ $\Delta = 1$ is the flat level, where the problem is convex
2: while $\Delta \neq 0$ do                          ▷ continue decreasing $\Delta$ until it is zero
3:     $\Delta \leftarrow \frac{\Delta}{f}$          ▷ f is a positive factor, which is determined experimentally
4:     Compute the minimizer of $S_\Delta(x) + \sqrt{1/2 * x' * H * x}$, denoted by $xmin$ using $z$ as a starting point
5:     $z \leftarrow xmin$
6: end while
7: return $xmin$                                    ▷ The output minimizer is $xmin$
```

Instead of trying to find the minimizer of the function directly, we break the problem into several levels and try to find the minimizer in each level using the minimizer found in the previous level as a starting point. We apply the build-in function $fmincon$ in MatLab for each level, and for the first level (i.e. the flat level), since $S_\Delta(x) = 0$ when $\Delta = 0$, we have a convex problem with a global minimizer, so we apply the function *quadprog* to find the minimizer of $\sqrt{1/2 * x' * H * x}$ using quadratic programming, where $x \geq 0$ and $Ax = b$.

The following figures illustrate how the smoothing method different from the smooth method, where del denotes δ and $\delta = \frac{(k-\Delta)}{k} * (level2 - level1)$.



(a) Smooth Method



(b) Smoothing Technique

Our algorithm starts from the flat level, where $\delta = 0$, that is $\Delta = 1$.

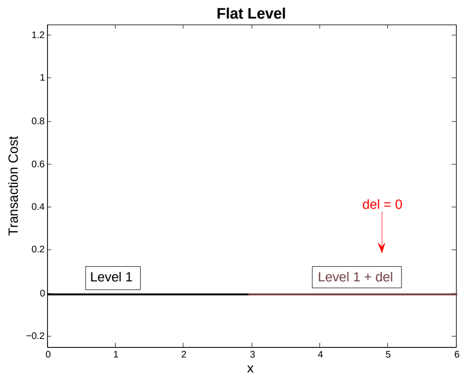


Figure 8: Flat Level

Since we have level 1 to be zero, at this level, the term $S_\Delta(x)$ is equal to zero. Thus, we only have the term $\sqrt{1/2 * x' * H * x}$, which is smooth and convex, and hence, there is a global minimizer of the function. We use the quadratic programming method to find this global minimizer, denoted by z, which can be done by applying the MatLab build-in function *quadprog*.

Then, we make an increment on level 1 by δ, which can be done by decreasing Δ, and use the z we found as a starting point for this level, and apply the MatLab build-in function $fmincon$ to find the minimizer for this level. For this level, we cannot use the $quadprog$, because $S_\Delta(x)$ is no longer equal to zero, and hence, the function is not a quadratic programming problem. And we denote the minimizer in this level as $xmin$.

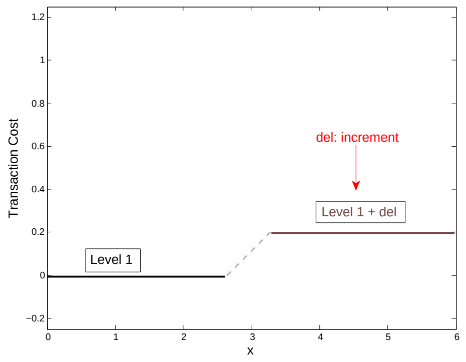


Figure 9: Next Level

And we continue to make increments by δ and move to next level until we reach level 2.

4. Numerical Results

In this section, we will present some examples of different variables and constraints, and we will compare the results from the smooth method and the smoothing method and show that the smoothing technique provides better results in most of the cases.

We will show results in $n = 4, 5, 6, 13, 15, 25, 50, 75, 80, 98$ dimensions. For each of the dimension, we first randomly generate the H matrix, which is positive definite and is of the size $n \times n$. After we generate the H, we keep it fixed for each dimension.

Then we will randomly generate the constraint matrices A and b as well as the starting points using the MatLab build-in function *randn*.

We present some results in the following tables, which list the function values computed and the smooth method, and the better results are bolded.

For $n = 4$:

A=[1,1,1,1,1,1;1.2741,0.9577,0.9796,0.9517,1.0638,1.0626]; b=[1;1.2];				
Starting Points	Smoothing Technique Function Value	Minimizer (computed by new method)	Local Method Function Value	Minimizer (computed by local method)
[-0.6312; -0.5003; -0.8672; -1.0401; 1.2654; -0.2415]	**0.1808**	[0.6476; 0.0000; 0.0000; 0.0000; 0.3523; 0.0000]	**0.1808**	[0.6476; 0; 0; 0; 0.3524; 0]
[$10^{11} \times 6.8630$; $10^{11} \times 0.3200$; $10^{11} \times -0.2564$; $10^{11} \times -2.1142$; $10^{11} \times -1.7769$; $10^{11} \times -6.3918$]	**0.1808**	[0.6476; 0; 0; 0; 0.3523; 0]	2.2575×10^{11}	[$10^{11} \times 6.863$; $10^{11} \times 0.320$; 0; 0; 0; 0]
$10^{11} \times$[4.0397; -2.3685; -2.7481; -0.0324; 1.7365; 0.0042]	**0.1808**	[0.6476; 0.0000; 0.0000; 0.0000; 0.3523; 0.0000]	1.0914×10^{11}	$10^{11} \times$[4.0397; 0; 0; 0; 1.7365; 0.0042]
$10^{11} \times$ [0.7196; 1.4103; 5.8768; -9.3534; 2.5614; -1.2339]	**0.1808**	[0.6476; 0.0000; 0.0000; 0.0000; 0.3523; 0.0000]	1.474211	$10^{11} \times$[0; 0; 2.0684; 0.8732; 1.5049; 1.4197]
$10^{11} \times$ [0.0000; 0.0000; 0.4814; 4.4447; 3.4848; 3.2019]	**0.1808**	[0.6476; 0.0000; 0.0000; 0.0000; 0.3523; 0.0000]	3.5010×10^{11}	$10^{11} \times$ [0; 0; 0.4814; 4.4447; 3.4848; 3.2019]
$10^{11} \times$ [3.4904; -0.6770; 0.2106; -1.2010; -0.1281; 1.5111]	**0.1808**	[0.6476; 0.0000; 0.0000; 0.0000; 0.3523; 0.0000]	1.5973×10^{3}	$10^{3} \times$ [3.9564; 1.1849; 0.0056; 0.0003; 0; 0.0015]
[26.3024; -5.8052; 82.7728; 162.6502; 2.1088; -19.0014]	**0.1808**	[0.6476; 0.0000; 0.0000; 0.0000; 0.3523; 0.0000]	**0.1808**	[0.6476; 0; 0; 0; 0.3524; 0]

For $n = 5$:

A=[1,1,1,1,1,1,1;0.7346,0.7118,1.0804,1.294,0.9346,1.1625,1.1091]; b=[1;1.2];				
Starting Points	Smoothing Technique Function Value	Minimizer (computed by new method)	Local Method Function Value	Minimizer (computed by local method)
$10^{12}\times$[0.4358, -1.0633, 0.1880, 0.6217, 0.1638, 0.6765, 1.5680]	**0.1590**	[0.0000, 0.0000, 0.1447, 0.4935, 0.0359, 0.1886, 0.1373]	131.7166	[0.0512, 0.0247, 438.1388, 36.3353, 0.0554, 0.1320, 0.0006]
$10^{12}\times$[0.2255, 0.5421, 0.3595, -1.1108, -0.3518, 1.0347,-0.0368]	**0.1590**	[0.0000, 0.0000, 0.1447, 0.4935, 0.0359, 0.1886, 0.1373]	4.7587 $*$ 10^7	$10^7\times$[0.0000, 5.7481, 0.0000, 1.0494, 0.0978, 7.5518, 0.0000]
$10^{11}\times$[0.4889, 0.5086, -0.3344, 0.8237, 0.5190, 1.9708, 0.8354]	**0.1590**	[0.0000, 0.0000, 0.1447, 0.4935, 0.0359, 0.1886, 0.1373]	3.0025 $*$ 10^9	$10^9\times$[0.0082, 0.0000, 8.4218, 3.0048, 8.9139, 1.7600, 5.8724]
$10^{10}\times$[-7.4149, -9.5620, 0.4023, -2.7776, -2.4855, -9.1918, 5.2651]	**0.1590**	[0.0000, 0.0000, 0.1447, 0.4935, 0.0359, 0.1886, 0.1373]	**0.1590**	[0.0000, 0.0000, 0.1447, 0.4935, 0.0359, 0.1886, 0.1373]
$10^9\times$[3.0308, -2.7009, -4.9916, 3.7788, -0.1712, 1.2253, -0.9915]	**0.1590**	[0.0000, 0.0000, 0.1447, 0.4935, 0.0359, 0.1886, 0.1373]	9.2251 $*$ 10^7	$10^8\times$[0.0000, 0.4900, 0.0009, 2.4002, 2.6791, 0.1760, 0.5496]

For $n = 6$:

A=[1,1,1,1,1,1,1,1; 1.0492,1.0140,0.8783,0.7555,1.0633,0.7314,0.7936,1.2662]; b=[1;1.2];				
Starting Points	Smoothing Technique Function Value	Minimizer (computed by new method)	Local Method Function Value	Minimizer (computed by local method)
$10^{10}\times$[-0.6665, -0.7842, 0.3746, 1.0707, -1.1584, -0.3074, -0.0577, -1.3627]	**0.3029**	[0.3051, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.6949]	4.5971 $*$ 10^9	$10^{10}\times$[0.0000, 0.0000, 0.3746, 1.0707, 0.0000, 0.0000, 0.0000, 0.0000]
$10^{10}\times$[-0.2861, 0.3893, 1.1282, 1.3321, -0.0866, -0.2372, 0.6055, 0.3630]	**0.3029**	[0.3051, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.6949]	1.5085e $*$ 10^8	$10^8\times$[0.0000, 1.9653, 0.0000, 0.0000, 0.0000, 1.9643, 0.0000, 0.0000]
$10^{11}\times$[1.2730, -2.4213, 0.8839, -0.1497, -1.2435, -1.1345, 2.0961, -1.3245]	**0.3029**	[0.3051, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.6949]	3.2366 $*$ 10^{10}	$10^{10}\times$[0.0425, 4.1049, 0.0000, 0.0006, 0.0003, 0.0000, 5.5810, 0.0000]
$10^{10}\times$[0.7850, 7.3520, 0.0934, 1.1272, -3.4305, 6.1214, 0.4570, 1.9382]	**0.3029**	[0.3051, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.6949]	**0.3029**	[0.3051, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.6949]
$10^{13}\times$[1.5436, 0.7033, -2.0893, 0.0718, -0.8144. 0.1975, -0.4528, -0.3432]	**0.3029**	[0.3051, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.6949]	4.5220 $*$ 10^{11}	$10^{11}\times$[0.0000, 1.6050, 0.0000, 0.2201, 0.2395, 0.0000, 6.8892, 5.7357]

For $n = 13$:

Starting Points Index	Smoothing Technique Function Value	Local Method Function Value
1	**0.0852**	$1.2572 * 10^8$
2	**0.0852**	$2.5256 * 10^8$
3	**0.0852**	$2.4341 * 10^7$
4	**0.0852**	$1.4822 * 10^9$
5	**0.0852**	$3.2170 * 10^7$
6	**0.0852**	$1.0043 * 10^9$
7	**0.0852**	$2.7345 * 10^8$
8	**0.0852**	**0.0852**
9	**0.0852**	$8.2846 * 10^5$
10	**0.0852**	$1.4652 * 10^8$
11	**0.0852**	$1.9325 * 10^9$
12	**0.0852**	$2.0372 * 10^8$
13	**0.0852**	$4.4882 * 10^7$
14	**0.0852**	$1.4948 * 10^8$

For $n = 15$:

Starting Points Index	Smoothing Technique Function Value	Local Method Function Value
1	**1.1452**	$2.5037 * 10^8$
2	**1.1452**	$3.8899 * 10^9$
3	**1.1452**	$3.6806 * 10^9$
4	**1.1452**	$4.2885 * 10^9$
5	**1.1452**	$4.9499 * 10^{11}$
6	**1.1452**	$2.8274 * 10^{11}$
7	**1.1452**	$7.6344 * 10^{11}$
8	**1.1452**	$1.2937 * 10^{11}$
9	**1.1452**	$3.3868 * 10^{12}$
10	**1.1452**	$5.1544 * 10^{12}$
11	**1.1452**	$1.2371 * 10^{12}$
12	**1.1452**	$7.3769 * 10^{11}$
13	**1.1452**	$7.96842 * 10^{12}$
14	**1.1452**	$6.4084 * 10^{11}$

For $n = 25$:

Starting Points Index	Smoothing Technique Function Value	Local Method Function Value
1	**0.1427**	$5.1364 * 10^8$
2	**0.1427**	$8.2071 * 10^8$
3	**0.1427**	$8.0994 * 10^7$
4	**0.1427**	$2.0008 * 10^9$
5	**0.1427**	$2.6810 * 10^4$
6	**0.1427**	0.1434
7	**0.1427**	0.1508
8	**0.1427**	0.1435
9	**0.1427**	$6.5344 * 10^5$
10	**0.1427**	0.1466
11	**0.1427**	0.1433
12	**0.1427**	$2.1615 * 10^{10}$
13	**0.1427**	$6.8256 * 10^9$
14	**0.1427**	$1.1519 * 10^{10}$

For $n = 50$:

Starting Points Index	Smoothing Technique Function Value	Local Method Function Value
1	**0.0852**	$4.3792 * 10^7$
2	**0.0852**	$2.1447 * 10^9$
3	**0.0852**	$5.6096 * 10^8$
4	**0.0852**	$1.3468 * 10^8$
5	**0.0852**	$1.7114 * 10^9$
6	**0.0852**	0.0916
7	**0.0852**	0.0899
8	**0.0852**	0.1160
9	**0.0852**	0.0915
10	**0.0852**	$2.4477 * 10^6$
11	**0.0852**	$2.1198 * 10^6$
12	**0.0852**	$1.8307 * 10^6$
13	**0.0852**	$2.3044 * 10^6$
14	**0.0852**	$1.5957 * 10^6$

For $n = 75$:

Starting Points Index	Smoothing Technique Function Value	Local Method Function Value
1	**2.9078**	$1.8717 * 10^9$
2	**2.9078**	$2.7826 * 10^4$
3	**2.9078**	$1.9388 * 10^4$
4	**2.9078**	2.9205
5	**2.9078**	$7.8828 * 10^4$
6	**2.9078**	2.9083
7	**2.9078**	2.9206
8	**2.9078**	$4.0374 * 10^5$
9	**2.9078**	$5.7569 * 10^5$
10	**2.9078**	$1.1771 * 10^5$
11	**2.9078**	$1.9755 * 10^5$
12	**2.9078**	$2.0452 * 10^9$
13	**2.9078**	$2.6688 * 10^{10}$
14	**2.9078**	$5.1237 * 10^{10}$

For $n = 80$:

Starting Points Index	Smoothing Technique Function Value	Local Method Function Value
1	**2.9789**	2.9922
2	**2.9789**	3.1159
3	**2.9789**	3.1984
4	**2.9789**	3.0103
5	**2.9789**	$1.0168 * 10^6$
6	**2.9789**	$1.4685 * 10^5$
7	**2.9789**	$2.0015 * 10^6$
8	**2.9789**	$8.8167 * 10^8$
9	**2.9789**	$5.1380 * 10^8$
10	**2.9789**	$3.8031 * 10^{11}$
11	**2.9789**	$3.2858 * 10^{11}$
12	**2.9789**	$6.7052 * 10^{10}$
13	**2.9789**	$3.8103 * 10^9$
14	**2.9789**	$3.7978 * 10^{10}$

Starting Point Index

For $n = 98$:

Starting Points Index	Smoothing Technique Function Value	Local Method Function Value
1	**3.2143**	88.5459
2	**3.2143**	105.2948
3	**3.2143**	114.7464
4	**3.2143**	494.5187
5	**3.2143**	257.19476
6	**3.2143**	249.7375
7	**3.2143**	3.2179
8	**3.2143**	$3.4415 * 10^6$
9	**3.2143**	$1.1419 * 10^6$
10	**3.2143**	$2.8539 * 10^6$
11	**3.2143**	$1.4312 * 10^{10}$
12	**3.2143**	$1.5532 * 10^{10}$
13	**3.2143**	$1.2537 * 10^{10}$
14	**3.2143**	$1.7907 * 10^{10}$

5. Conclusion

In this paper, we consider a minimization problem with piecewise constant transaction costs, which is a major problem in financial portfolio allocation. We develop and apply a new smoothing technique to solve this problem.

The piecewise constant transaction cost makes the problem hard to solve by smooth minimization methods due to its discontinuity. Thus, we develop a smoothing technique to solve this kind of problems, which can be formulated as $min\{S_\Delta(x) + \sqrt{1/2 * x' * H * x} : Ax = b, x \geq 0\}$, where the latter term $\sqrt{1/2 * x' * H * x}$ represents risk in finance.

We begin the process with the flat level, i.e. $\Delta = 1$, which is a convex problem and has a global minimizer. We then solve a sequence of local minimization problems, where each problem is defined by parameter Δ and Δ is reduced from unity to zero. The starting point for a local minimization process in this sequence is the solution to the previous local minimization problem in the sequence. Hence, we are tracking a sequence of solutions starting from the global minimizer of a convex problem (i.e., when $\Delta = 1$).

Our numerical results support the claim that this is an effective approach to the portfolio allocation problem with piecewise-constant transaction costs.

We notice that if we start from a point x where $TCb(x)$ equals to $level1$ or $level2$, then this point is a local minimizer, and hence, our methods are unable to make any improvements. However, we can fix this by letting Δ be large enough so that most of the starting points are lying in the set of the linear interpolant.

References:

[1] Xi, Coleman, Li, and Tayal (2012), A Gradual Non-Convexation Method for Minimizing VaR.

[2] Coleman, and Li, A Graduated Nonconvex Regularization for Sparse High Dimensional Model Estimation. *Journal of Computer and Communications,* $2014, 2, 1 - 14$.

[3] S.J. Wright and J. Nocedal. Numerical Optimization. Second edition, 2004.

[4] M.J.Best. Portfolio Optimization. 2010.