A Technique for Choosing an Effective Hedging Portfolio with Few Instruments

by

Varuna Manevannan

An essay
presented to the University of Waterloo
in partial fulfillment of the
requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Prof. Thomas F. Coleman

Waterloo, Ontario, Canada, 2017

I hereby declare that I am the sole author of this report. This is a true copy of the report, including any required final revisions, as accepted by my examiners.

I understand that my report may be made electronically available to the public.

Abstract

This essay explores the sensitivity of a financial portfolio to the number of active instruments in it while maintaining risk protection. The transaction costs of the options should not outweigh the expected risk of the underlying security. Therefore, it is important to identify the instruments that can cost-effectively insure the associated risk. However, the transaction cost function is discontinuous resulting in a non-convex optimization problem making it NP-hard. We investigate solution methods using the following framework. Firstly, the counting function in the objective is approximated by an alternative smoothed function. Then the resulting formulation is optimized using an interior-point algorithm which runs a local solver from multiple starting points.

Acknowledgements

I would like to thank my advisor Dr Thomas Coleman for his supervision and patience. He motivated me to learn a lot this year. Thanks to my reader Dr George Labahn for his comments and insightful feedback.

I thank Su Yan [15] for sharing her work and Cayuga Research, the developers of ADMAT. I owe my gratitude to the CM department for facilitating an incredible learning environment.

Thanks to the Settlers of Catan and those in MC 6011. I would like to thank my parents, thatha, maadi, Akshay, Shashi, and Ginger for their unconditional support.

To Sai Mali for motivating me to do what I need to do even when I didn't feel like it.

Dedication

To my parents, Kavitha and Manevannan.

Table of Contents

List of Figures

Chapter 1

Introduction

Hedging is a risk-management strategy to combat adverse price movements. The securities that comprise the hedge are picked such that they move in a different direction than the rest of the portfolio, appreciating when the other instruments decline and vice-versa.

Occasionally the market experiences volatility and this has a bearing on portfolio values due to stock price movements, interest rate fluctuation and currency exchange swings. To match these wide variety of risks that a market poses, there exists a large assortment of instruments such as options and futures to hedge these risks. Options make for very attractive hedges because they can be easily leveraged. In this essay, we construct a hedging portfolio that is comprised of simple European call and put options.

Hedging provides a solid floor in the event of a catastrophe. It compares closely to insurance i.e., it provides protection but comes at a cost including transaction costs.

There exist several ways to model transaction costs, namely as a percentage of the underlying asset, a fixed cost for each share or a flat fee for each trade. In this essay, we focus on modelling a suitable transaction cost function where we associate a cost proportional to the number of nonzero instruments in a portfolio.

When transaction costs are ignored the result is a convex quadratic programming problem and hence easy to solve; however the resulting solution may be unaffordable in practice. Including transaction costs provides a more accurate description of the practical investment conditions.

The primary challenges posed by this transaction cost function in an optimization setting are non-convexity, existence of multiple minima and NP-hardness. Therefore, heuristic procedures have been developed to approximately solve the portfolio hedging problem with transaction costs. In this essay we propose an efficient and effective approach by means of a graduated minimization technique using a smoothed transaction cost function.

The organization of the rest of this essay is as follows. Chapter 2 provides some preliminaries and discusses two optimization strategies. Chapter 3 discusses our main result, which is a smoothed approach for transaction costs. In Chapter 4, we illustrate our method with computational results.

Chapter 2

Preliminaries and Problem Description

Definition 1 (European Call Option) *A European call option C gives the holder the right but not the obligation to buy an underlying asset S_T at a particular time period T for a certain strike price K.*

Definition 2 (European Put Option) *A European put option P gives the holder the right but not the obligation to sell an underlying asset S_T at a particular time period T for a certain strike price K.*

Their payoffs are described as follows:

$$C(S,T) = \max(S_T - K, 0)$$

$$P(S,T) = \max(K - S_T, 0)$$

where T is the time to maturity.

The value of the contract is given by $V(S,t)$.

The Black-Scholes-Merton formulas [3] for pricing European call and put options are as follows:

$$C(S,t) = S_t\Phi(d_1) - Ke^{-r\tau}\Phi(d_2)$$
$$P(S,t) = Ke^{-r\tau}\Phi(-d_2) - S_t\Phi(-d_1)$$

where

$$
\begin{aligned}
d_1 &= \frac{ln(\frac{S_t}{K}) + (r + \sigma^2/2)\tau}{\sigma\sqrt{\tau}} \\
d_2 &= \frac{ln(\frac{S_t}{K}) + (r - \sigma^2/2)\tau}{\sigma\sqrt{\tau}} = d_1 - \sigma\sqrt{\tau} \\
\tau &= T - t
\end{aligned}
$$

The function $\Phi(x)$ is the cumulative probability distribution function for a standardized normal distribution.

Let us construct a portfolio comprising d risk factors (e.g. stocks) $\mathbf{S} = (s_1, s_2, \ldots, s_d)$ and n simple options $\mathbf{V} = (V_1(\mathbf{S}, t), ..., V_n(\mathbf{S}, t))^T$ for hedging purposes. A simple option is one that depends exactly on a single underlying asset s_j. The vector $\mathbf{V}$ contains simple options V_i that are either call (Definition 1) or put (Definition 2) options.

The hedging portfolio is given by:

$$\Pi(\mathbf{x}, \mathbf{S}, t) = \sum_{i=1}^{n} x_i V_i(\mathbf{S}, t) \tag{2.1}$$

where $\mathbf{x} = (x_1, ..., x_n)^T$ and x_i is the investment on option V_i.

Let $\Pi_0(\mathbf{S}, t)$ be the portfolio to be hedged at time $t = 0$. The gradient $\mathbf{g} \in \mathbb{R}^{d+1}$ of Π_0 is denoted as:

$$\mathbf{g} = \begin{bmatrix} \dfrac{\partial \Pi_0}{\partial t} \\ \\ \nabla_{\mathbf{S}} \Pi_0 \end{bmatrix}. \tag{2.2}$$

The Jacobian $J_{\mathbf{V}} \in \mathbb{R}^{n \times (d+1)}$ of the vector of hedging instruments $\mathbf{V}$ is denoted as:

$$J_{\mathbf{V}} = \begin{bmatrix} \frac{\partial V_1}{\partial t} & \frac{\partial V_1}{\partial s_1} & \cdots & \frac{\partial V_1}{\partial s_d} \\ \frac{\partial V_2}{\partial t} & \frac{\partial V_2}{\partial s_1} & \cdots & \frac{\partial V_2}{\partial s_d} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial V_n}{\partial t} & \frac{\partial V_n}{\partial s_1} & \cdots & \frac{\partial V_n}{\partial s_d} \end{bmatrix}. \tag{2.3}$$

2.1 Exploiting the Structure of the Jacobian

Since our hedging portfolio uses only simple European options, each hedging instrument V_i is a function of exactly one element of $\mathbf{S}$, so each row of $J_{\mathbf{V}}$ has exactly two non-zero values.

The first non-zero value is $\dfrac{\partial V_i}{\partial t}$ and the other non-zero value $\dfrac{\partial V_i}{\partial s_j}$ for a unique j lies in a column between $2 : d+1$. The Jacobian is efficiently evaluated by exploiting its sparsity structure [9].

We compute the Jacobian by applying forward mode Automatic Differentiation (Section A.1) to the vector function $\mathbf{V}(t, \mathbf{S})$. In fact, we only need two passes of the Automatic Differentiation (AD) tool. By the first pass, we can compute $J_{\mathbf{V}}\mathbf{e}_1$, where $\mathbf{e}_1$ is the first column of a $(d+1) \times (d+1)$ identity matrix, this determines the first column of the Jacobian matrix $J_{\mathbf{V}}$. Using the second pass, we can compute $J_{\mathbf{V}}\overline{\mathbf{e}}_1$, where $\overline{\mathbf{e}}_1 = (0, 1, 1, \ldots, 1)^T$, which then yields all the non-zeros of columns $2 : d+1$ of $J_{\mathbf{V}}$.

2.2 Optimization Strategies by Hedging Greeks

By expressing the change in portfolio value $\Pi(S, t)$ in terms of a Taylor series we know that the value $\Pi(S, t)$ of a portfolio satisfies the following differential equation [10]:

$$\frac{\partial \Pi}{\partial t} + rS\frac{\partial \Pi}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 \Pi}{\partial S^2} = r\Pi. \tag{2.4}$$

Define the following Greeks (Appendix A.2),

$$\Theta = \frac{\partial \Pi}{\partial t}, \quad \Delta = \frac{\partial \Pi}{\partial S}, \quad \Gamma = \frac{\partial^2 \Pi}{\partial S^2}. \tag{2.5}$$

Note: In practice σ is often defined as $\sigma(S, t)$ so one obtains a simple extension of (2.4). When σ is not constant and instead a surface that varies with S and t, the analytical formulas for Greeks are not readily available. This situation demands for Automatic Differentiation to be used in order to compute the values in (2.5).

It follows that

$$\Theta + rS\Delta + \frac{1}{2}\sigma^2 S^2 \Gamma = r\Pi.$$

For a delta-neutral portfolio we have $\Delta = 0$ therefore,

$$\Theta + \frac{1}{2}\sigma^2 S^2 \Gamma = r\Pi. \tag{2.6}$$

Equation (2.6) provides a closed form expression for theta in terms of gamma. This explains why theta may be regarded as a proxy for gamma in a delta-neutral portfolio.

The risk protection approach involves minimizing a quadratic objective function subject to linear constraints. We construct the problem formulation in sections 2.2.1 and 2.2.2 keeping the following details in mind:

The first strategy neutralizes delta and theta. A delta hedge captures the sensitivity of a portfolio to the underlying via a linear approximation. The theta measure is included since it is a substitute for gamma in a delta-neutral portfolio. In practice, we observe that the principal method of dynamic hedging is Delta hedging because the first order measures, delta and theta represent the primary component of change in option value.

In the second strategy we employ the Greek, gamma. It improves hedging effectiveness by allowing for larger interval trading.

2.2.1 First Optimization Strategy: Delta-Theta Approach

Consider the hedged portfolio from (2.1),

$$\Pi_0(\mathbf{S}, t) - \Pi(\mathbf{x}, \mathbf{S}, t) = \Pi_0(\mathbf{S}, t) - \sum_{i=1}^{n} x_i V_i(\mathbf{S}, t). \tag{2.7}$$

A delta-theta approach to hedging at time $t = 0$ is to choose a hedge strategy $\mathbf{x}$ such that the first-order change in (2.7), wrt $[t; \mathbf{S}]$ is zero, i.e.,

$$\mathbf{g} - J_{\mathbf{V}}^T \mathbf{x} = 0. \tag{2.8}$$

But (2.8) is underdetermined and admits many solutions, some of which may be unreasonable due to practical limitations. In that light, we will now account for the cost of the

hedge in (2.8). Assuming it costs c_i to acquire 1 unit of V_i while allowing no shorting of the hedge instruments, we now have the following hedging approach:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \left\| \mathbf{g} - J_{\mathbf{V}}^T \mathbf{x} \right\|_2 + \mathbf{c}^T \mathbf{x} \\ \text{subject to} \quad & \mathbf{0} \leq \mathbf{x} \leq \mathbf{u} \end{aligned} \tag{2.9}$$

where $\mathbf{c} = (c_1, c_2, \ldots, c_n)^T$ is cost and $\mathbf{u} = (u_1, u_2, \ldots, u_n)^T$ is a positive vector of upper bounds i.e., investing in the corresponding hedging instruments costs $\mathbf{c}$ and is bounded above by $\mathbf{u}$. In order to set this up as a constrained convex problem, we assume a total hedging budget W. We now have the following formulation:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \left\| \mathbf{g} - J_{\mathbf{V}}^T \mathbf{x} \right\|_2^2 \\ \text{subject to} \quad & \mathbf{c}^T \mathbf{x} \leq W \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}. \end{aligned} \tag{2.10}$$

We solve (2.10) for a range of wealth constraints $\mathbf{c}^T \mathbf{x} \leq W$. By varying the values of W, we obtain a frontier of hedges ranging from no hedge ($W = 0$) to a complete hedge (say some value $W_{\max}$).

Notice that Problem (2.10) is a convex quadratic program (QP) and hence is easy to solve. In the objective, we have $\mathbf{g} \in \mathbb{R}^{(d+1)}$ and $J_{\mathbf{V}}^T \in \mathbb{R}^{(d+1)\times n}$. For the unknown $\mathbf{x} \in \mathbb{R}^n$, we have $(2n+1)$ inequality constraints. For a linearly constrained convex QP, the number of iterations of a good interior point method is approximately $O(\sqrt{(2n+1)}\log(\frac{1}{\varepsilon}))$ with each iteration $O(n^2(2n+1))$ where ε is the error in computed value compared to actual value, $(2n+1)$ is the number of polyhedral constraints and n is the number of variables [16]. In our case, the matrix $J_{\mathbf{V}}$ is sparse and this further reduces the running time of each iteration of the QP.

2.2.2 Second Optimization Strategy: Gamma Approach

In order to increase the accuracy of the hedge we add second derivatives (gamma hedging). Since we are considering only simple options in the hedging portfolio, we know that each hedge instrument depends exactly on one risk factor. Therefore,

$$\frac{\partial^2 V_i}{\partial S_k \partial S_j} = 0 \quad \forall\ i, k \neq j.$$

Define

$$\Gamma_i = \left[\frac{\partial^2 V_i}{\partial S_1^2}, \ldots, \frac{\partial^2 V_i}{\partial S_d^2}\right]^T.$$

Note that exactly one component of Γ_i is nonzero. AD is once again tailored to get each Γ_i efficiently by exploiting its sparsity structure. We now extend (2.8) to match second derivatives as well. Specifically, define

$$\mathbf{\Gamma} = \begin{bmatrix} \Gamma_1^T \\ \Gamma_2^T \\ \vdots \\ \Gamma_n^T \end{bmatrix} \in \mathbb{R}^{n \times d} \tag{2.11}$$

and so $\mathbf{\Gamma}$ has exactly 1 nonzero in each row. Equation (2.8) now becomes,

$$\begin{bmatrix} \mathbf{g} \\ \mathbf{\Gamma}_{\Pi_0} \end{bmatrix} - \begin{bmatrix} J_{\mathbf{V}}^T \\ \Gamma^T \end{bmatrix} \mathbf{x} = 0. \tag{2.12}$$

The problem formulation now becomes:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \left\| \begin{bmatrix} \mathbf{g} \\ \mathbf{\Gamma}_{\Pi_0} \end{bmatrix} - \begin{bmatrix} J_{\mathbf{V}}^T \\ \Gamma^T \end{bmatrix} \mathbf{x} \right\|_2^2 \\ \text{subject to} \quad & \mathbf{c}^T \mathbf{x} \leq W \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{u} \end{aligned} \tag{2.13}$$

Similar to Delta-Theta approach, we solve (2.13) for a range of wealth constraints $\mathbf{c}^T\mathbf{x} \leq W$. Once again, by varying the values of W, we obtain a frontier of hedges ranging from no hedge ($W = 0$) to a complete hedge (say some value $W_{\max}$).

Problem (2.13) is also a convex quadratic program (QP) and hence is easy to solve (see our discussions below Problem (2.10)). Note that this problem is of larger size than Problem (2.10) due to the inclusion of second-order terms. In the objective, we have $\begin{bmatrix} \mathbf{g} \\ \mathbf{\Gamma}_{\Pi_0} \end{bmatrix} \in \mathbb{R}^{(2d+1)}$ and $\begin{bmatrix} J_{\mathbf{V}}^T \\ \Gamma^T \end{bmatrix} \in \mathbb{R}^{(2d+1)\times n}$.

Chapter 3

Third Optimization Strategy: Smoothed Approach for Transaction Costs

The solution to (2.10) or (2.13) could have many active hedging functions i.e., many non-zero components in the original hedge strategy $\mathbf{x}$. Transaction costs are assumed to be proportional to the number of active instruments $|\mathbf{x}|_{NNZ}$.

3.1 A Revised Portfolio Problem with Transaction Costs

The portfolio investment problem described in (2.13) has the following objective after including transaction costs:

$$f(\mathbf{x}) = \left\| \begin{bmatrix} \mathbf{g} \\ \mathbf{\Gamma}_{\Pi_0} \end{bmatrix} - \begin{bmatrix} J_{\mathbf{V}^T} \\ \Gamma^T \end{bmatrix} \mathbf{x} \right\|_2^2 + \omega . |\mathbf{x}|_{NNZ}$$

where $|\mathbf{x}|_{NNZ}$ is the number of active instruments in the portfolio and ω is the associated penalty cost. The new optimization problem is:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \left\| \begin{bmatrix} \mathbf{g} \\ \mathbf{\Gamma}_{\Pi_0} \end{bmatrix} - \begin{bmatrix} J_{\mathbf{V}^T} \\ \Gamma^T \end{bmatrix} \mathbf{x} \right\|_2^2 + \omega . |\mathbf{x}|_{NNZ} \\ \text{subject to} \quad & \mathbf{c}^T \mathbf{x} \leq W \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}. \end{aligned} \tag{3.1}$$

If $\omega = 0$, then there is no penalty imposed and this is the same as Problem (2.13).

3.2 A Smooth Approximation to the Counting Function

The term for transaction cost in the objective function of Problem (3.1) is neither continuous nor differentiable and solving the problem is NP-hard [2]. We perform the following approximation strategy to solve the problem.

Consider a piecewise constant function $I : \mathbb{R} \to \mathbb{R}$ to count the number of nonzeros of a vector $\mathbf{x}$. For a scalar input, this function is 1 when the argument is nonzero and 0 otherwise.

$$I(x) = \begin{cases} 1 & x \neq 0 \\ 0 & x = 0 \end{cases}$$

Figure 3.1 is a plot of $I(x)$. For a vector $\mathbf{x} \in \mathbb{R}^n$, we can define $I(\mathbf{x}) = \sum_{i=1}^{n} I(x_i)$. Thus $I(\mathbf{x})$ is the counting function and Problem (3.1) becomes

$$\begin{aligned} \min_{\mathbf{x}} \quad & \left\| \begin{bmatrix} \mathbf{g} \\ \mathbf{\Gamma}_{\Pi_0} \end{bmatrix} - \begin{bmatrix} J_{\mathbf{V}^T} \\ \Gamma^T \end{bmatrix} \mathbf{x} \right\|_2^2 + \omega I(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{c}^T \mathbf{x} \leq W \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{u} \end{aligned} \tag{3.2}$$

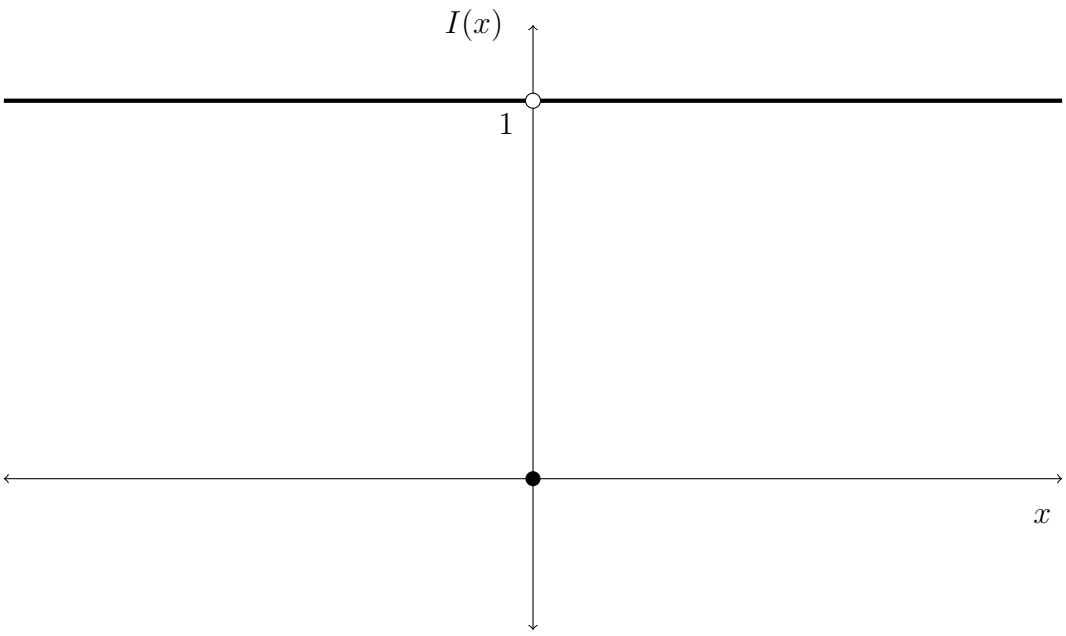

Figure 3.1: $I(x)$: Counting nonzeros.

In order to smooth this non-continuous function, we produce approximations to $I(x)$ as follows. We modify $I(x)$ in two stages via quadratic splines, in order to obtain a continuous and differentiable approximation. This is similar to the method in Coleman, Li and Henniger [8]. We call this smoothed approximation function $TC_\Delta : \mathbb{R} \to \mathbb{R}$, where Δ is a parameter controlling how close to zero the smoothing occurs.

At the first stage, we smooth the function at the origin and create a continuous but non-differentiable function. For $\Delta > 0$, let

$$Q_\Delta(x) = \begin{cases} 1 & |x| \geq \Delta \\ \frac{x^2}{\Delta^2} & \text{otherwise.} \end{cases} \tag{3.3}$$

Figure 3.2 is a plot of $Q_\Delta(x)$. The function is still not differentiable everywhere (see $x = \pm\Delta$). Notice that as $\Delta \to 0,\ Q_\Delta(x) \to I(x)$.

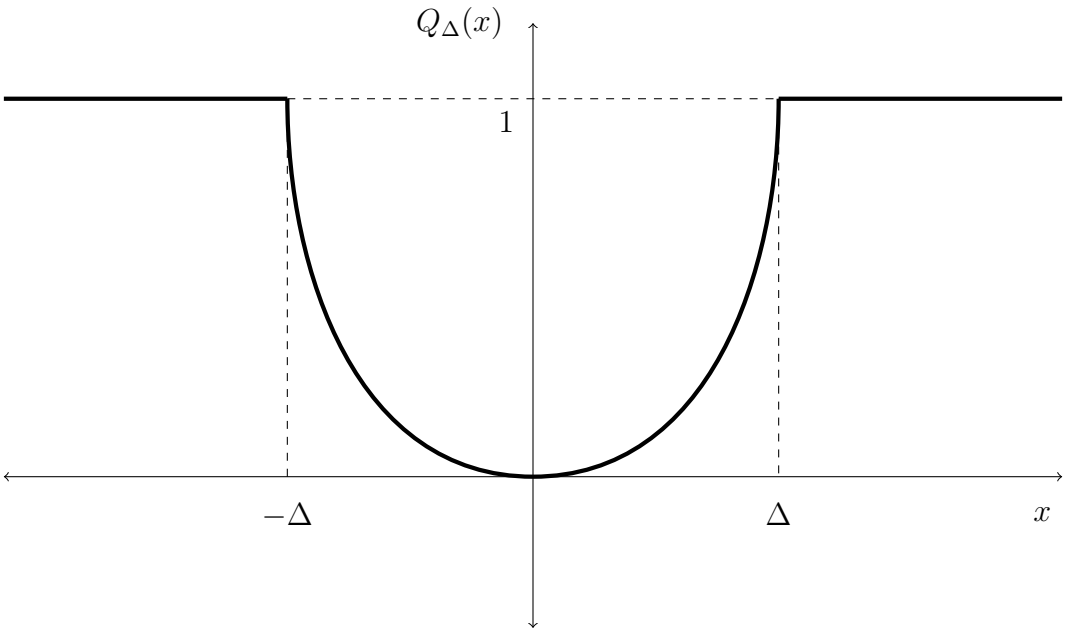

Figure 3.2: $Q_\Delta(x)$: Smoothing $I(x)$ at the origin.

Next, we smooth the points where the function Q_Δ was not differentiable.
For $\Delta \neq 0$,

$$TC_\Delta(x) = TC_\Delta(x) = \begin{cases} 1 & x \leq -\Delta - \varepsilon \\ 1 - \frac{m}{4\varepsilon}(x + \Delta + \varepsilon)^2 & -\Delta - \varepsilon < x \leq -\Delta + \varepsilon \\ \frac{1 - m\varepsilon}{(\Delta - \varepsilon)^2} x^2 & -\Delta + \varepsilon < x \leq \Delta - \varepsilon \\ 1 - \frac{m}{4\varepsilon}(x - \Delta - \varepsilon)^2 & \Delta - \varepsilon < x \leq \Delta + \varepsilon \\ 1 & x > \Delta + \varepsilon \end{cases} \tag{3.4}$$

where ε controls the length of quadratic spline approximation (we use $\varepsilon = \frac{\Delta}{10}$ and $m = \frac{20}{19\Delta}$). Section A.3 shows that TC_Δ is continuously differentiable.

For a vector $\mathbf{x} \in \mathbb{R}^n$, we define $TC_\Delta(\mathbf{x}) = \sum_{i=1}^n TC_\Delta(x_i)$. Figure 3.3 is a plot of the approximation function $TC_\Delta(x)$.

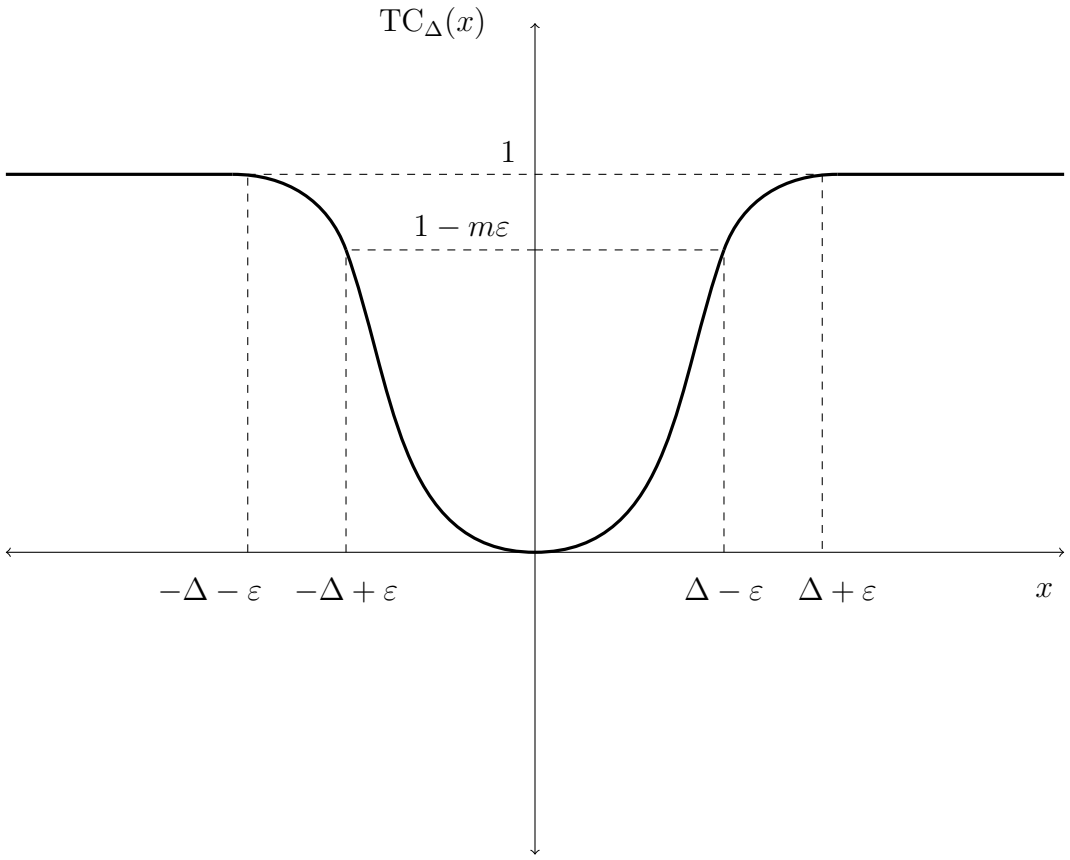

Figure 3.3: $TC_\Delta(x)$: A smoothed approximation function

3.3 Iterative Minimization with Smoothed Counting Function

On introducing the counting function to replace the second term in the objective of Problem (3.1), we are solving the minimization problem Problem (3.2) with $I(\mathbf{x})$ approximated by $TC_\Delta(\mathbf{x})$. If we replace the transaction cost term in Problem (3.2) with $TC_\Delta(\mathbf{x})$, the objective is a continuous, differentiable and nonconvex function that is piecewise quadratic.

As $\Delta \to 0$, $TC_\Delta(\mathbf{x})$ becomes a better approximation to $I(\mathbf{x})$. By reducing Δ sequentially, we use the following iterative algorithm to solve Problem (3.2).

Algorithm 1 Graduated minimization with smoothing technique

1: **Input**: The known quantities in Problem (3.2).
2: **Output**: Minimizer to Problem (3.2).
3: Find the global minimizer $\mathbf{z}$ of Problem (2.13) which is a convex QP.
    ▶ This is the optimizer of Problem (3.2) ignoring the term with $I(x)$ in the objective.
4: Set initial value of Δ, say $\Delta = 1$.
5: Set λ, say $\lambda = 10$. ▶ Positive factor found experimentally.
6: **while** $\Delta \neq 0$ **do** ▶ $\Delta \to 0$ gives better approximation to $I(\mathbf{x})$.
7: Solve the following problem with $\mathbf{z}$ as the starting point.

$$\begin{aligned} \mathbf{x}_{\min} = \arg\min_{\mathbf{x}} \quad & \left\| \begin{bmatrix} \mathbf{g} \\ \mathbf{\Gamma}_{\Pi_0} \end{bmatrix} - \begin{bmatrix} J_{\mathbf{V}^T} \\ \Gamma^T \end{bmatrix} \mathbf{x} \right\|_2^2 + \omega TC_\Delta(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{c}^T\mathbf{x} \leq W \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{u} \end{aligned} \tag{3.5}$$

▶ Solving an approximation to Problem (3.2).
8: Set $\mathbf{z} \leftarrow \mathbf{x}_{\min}$. ▶ Optimizer of current iteration is initial point of the next.
9: Set $\Delta \leftarrow \frac{\Delta}{\lambda}$. ▶ Decrease Δ for the next iteration.
10: **end while**
11: Return $\mathbf{x}_{\min}$.

From Algorithm 1, it seems like we solve a lot of approximation problems (3.5). But for many consecutive iterations, the starting point $\mathbf{z}$ (which was the minimizer of the previous iteration) also happens to be the minimizer of the current iteration [8]. Even if this is not the case, the minimizer $\mathbf{x}_{\min}$ of the previous iteration is nevertheless a good starting point for the current iteration. Thus the number of iterations in Algorithm 1 is not excessive.

Therefore, we need to solve the approximation problems (3.5) in the iterations of Algorithm 1. The objective of the optimization problem (3.5) is a continuous differentiable nonconvex function which need not have an unique minimum. Section A.4 provides details of the techniques to find the global optimizer of the approximation problem.

Chapter 4

Computational Results And Conclusion

4.1 Computational Results

For the numerical results, we implemented the Algorithm 1 using a global optimization method GLOBAL SEARCH(A.4) and the routine FMINCON from the MATLAB 2016*b*[13] Optimization Toolbox [12]. Through a MATLAB implementation of the optimization strategies described in Chapter 2.2 and Chapter 3 we have the following results:

A hedging portfolio of 100 European options consisting of 50 calls and 50 puts was considered. We obtained the following plots while testing 100 portfolios with varying budgets $W \in [1, 50]$.

- Delta-Theta Hedging Strategy 2.2.1

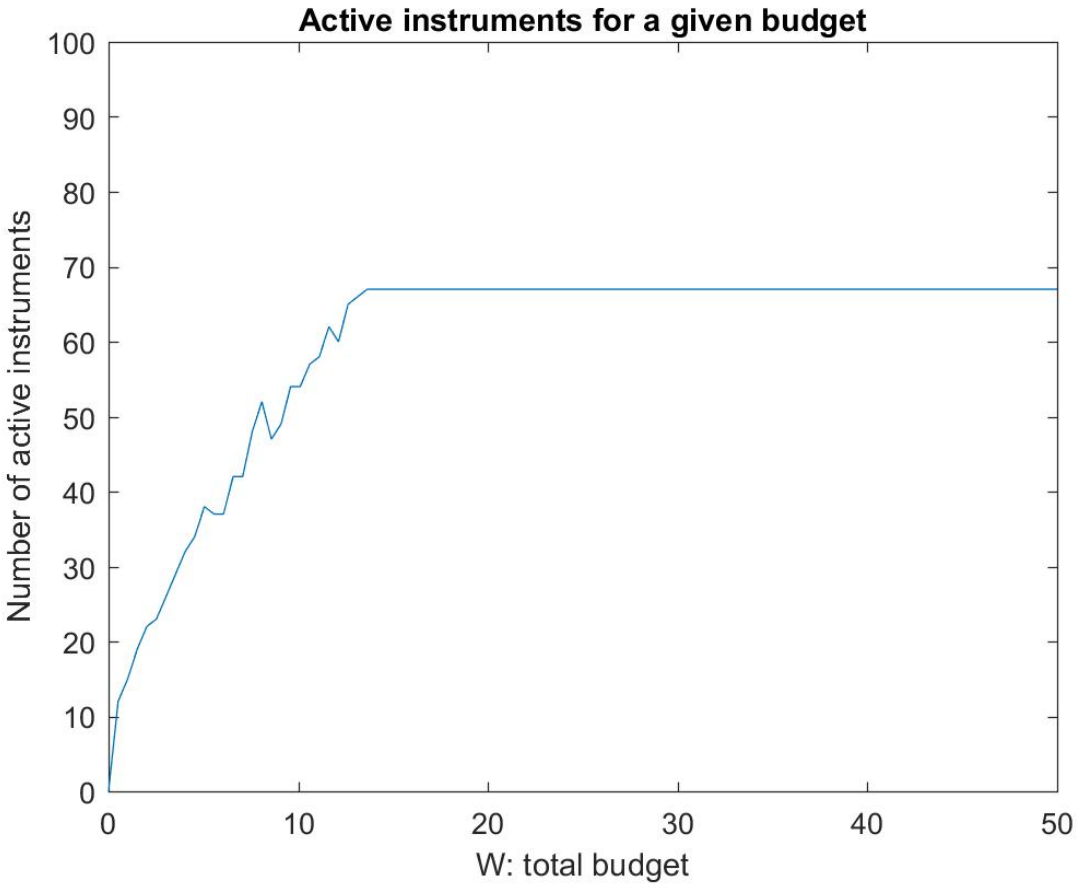

Figure 4.1: Number of active instruments in the portfolio for various total budgets.

Observations from Figure 4.1:

- Up to 70 out of 100 instruments became active while performing delta-theta hedging.
- The increasing trend in the graph supports the intuition that the number of the active instruments in an optimal portfolio increases with higher budget allocation. Thereby illustrating that a higher capital permits a more diversified portfolio.
- However, we also note that any increase beyond \$15 in the budget makes no difference to the number of active instruments in the portfolio.

- Gamma Hedging Strategy 2.2.2

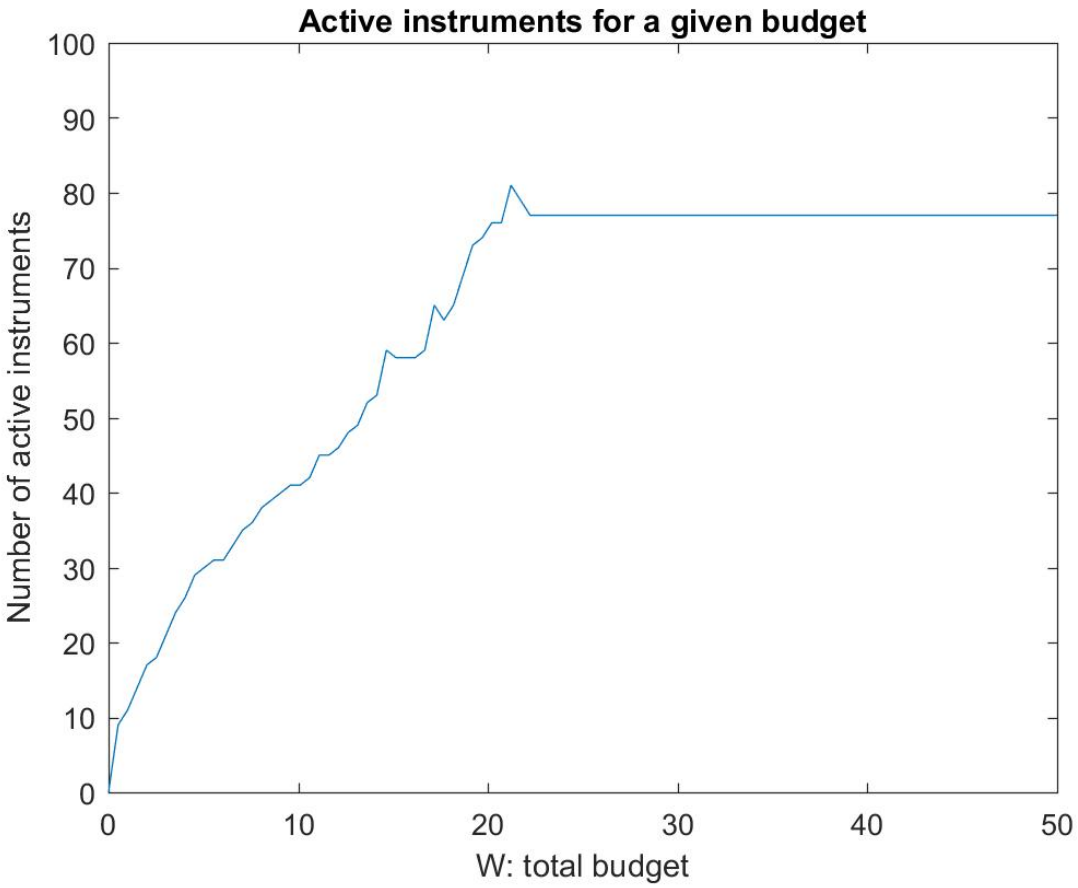

Figure 4.2: Number of active instruments in the portfolio for various total budgets.

Observations from Figure 4.2:

- Up to 80 out of 100 instruments became active while performing gamma hedging.
- We notice a steady increase in the number of active instruments with growing budget. Secondly, for the same budget we notice that the number of active instruments are higher here than the delta-theta trading strategy. Therefore,

in addition to higher capital we now know that second order hedging allows for greater diversification.

  - Similar to the delta-theta hedging, we note that any increase beyond \$22 in the budget makes no difference to the number of active instruments in the portfolio.

- Third Optimization Strategy: Smoothed approach for transaction costs, Chapter 3

  We fix a budget $W = 10$ and obtain the following plots.

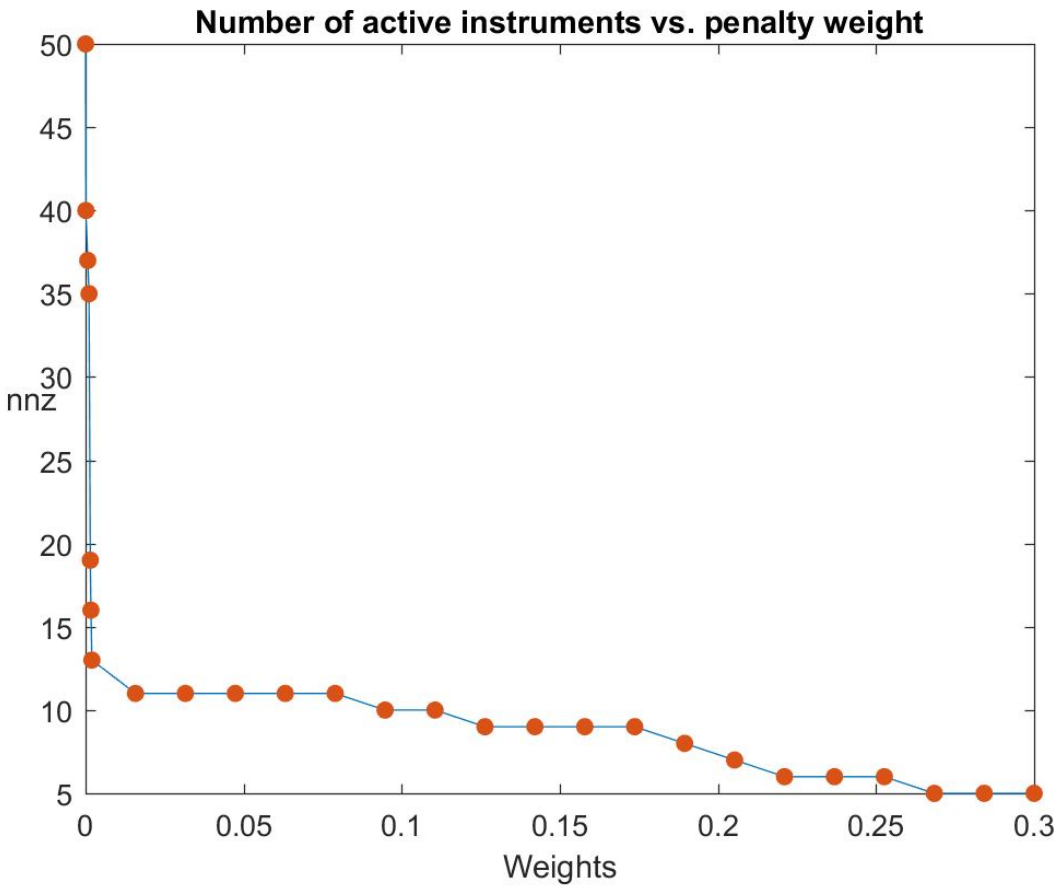

Figure 4.3: Number of active instruments in the portfolio for various penalty weights in the range $[0, 0.3]$

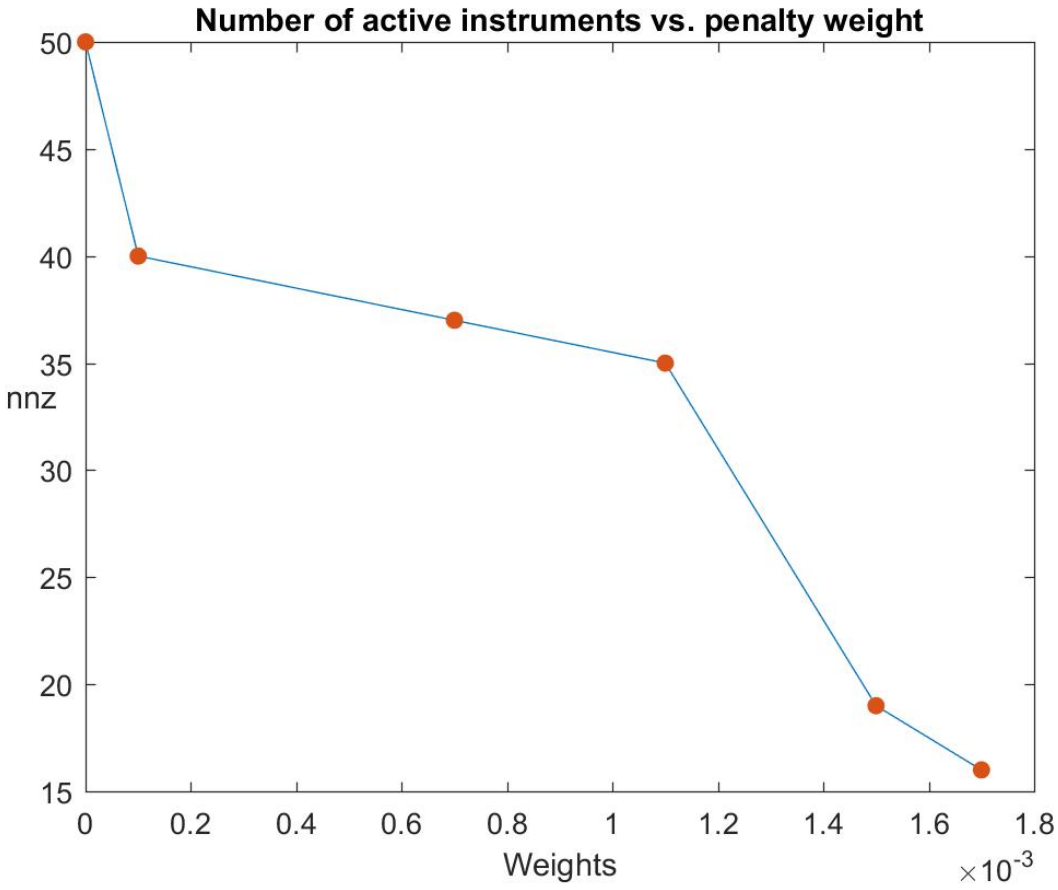

Figure 4.4: Number of active instruments in the portfolio for various penalty weights, for very small weights $[0, 1.8 \times 10^{-3}]$

Observations from Figure 4.3 and Figure 4.4:

- The portfolio had 50 active instruments when there was no transaction cost term ($\omega = 0$). On increasing the penalty imposed on transaction cost, 50 active instruments gradually declined to 5.

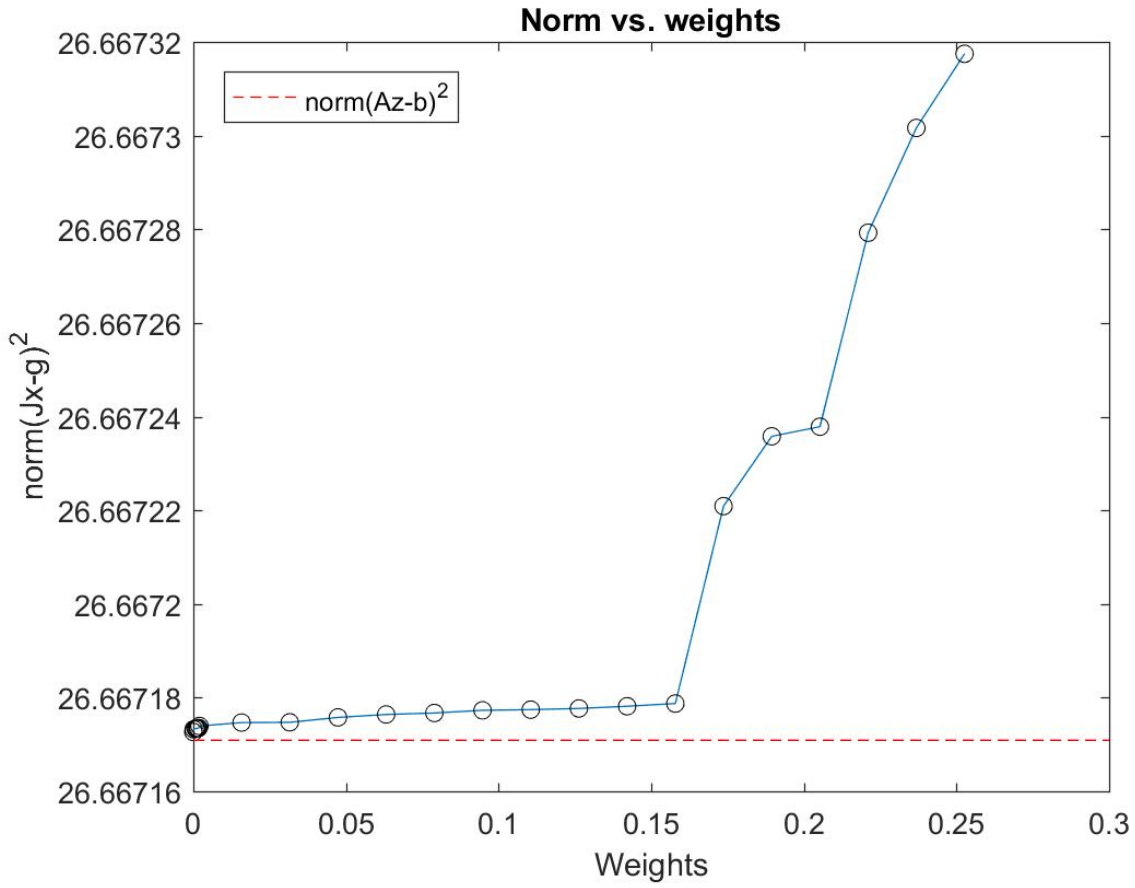

Figure 4.5: Value of norm term in the objective for the optimal portfolios for various penalty weights.

Observations from Figure 4.5:

- Even for a large problem involving 100 assets, numerical experiments suggest that the gap from the norm value of the convex problem (dashed line) is small.
- We observe that the norm term increases with increasing weight. This is expected since the optimizer for a higher weight is a candidate as an optimizer for a lower weight.

4.2 Conclusion

We developed a portfolio hedging technique that is sensitive to the number of active instruments. For hedging purposes, we assume a portfolio comprising of simple European options. We use Automatic Differentiation (ADMAT 2.0) to exploit the structural sparsity of Jacobians and Hessians to achieve more efficient computing time in computing derivatives for hedging purposes.

A practical hedging strategy takes transaction costs into account. Therefore, we formulate an optimization problem incorporating transaction costs. However this makes the objective function discontinuous and the problem was NP-hard. We model the counting function using a smoothed approximation which recovers the differentiability of the objective function, however leaving it nonconvex. We use a graduated minimization algorithm which iteratively solves the reformulated problem. Numerical results which implement the above technique illustrate that the number of active instruments can be reduced while maintaining the risk protection. It is noteworthy that instead of solving an NP-hard problem, we provide an approximate solution by solving a sequence of tractable optimization problems, which involve the smoothed approximation function.

APPENDICES

Appendix A

Background details

A.1 Using Automatic Differentiation to Obtain Derivatives

There exist various techniques for obtaining gradients and Hessians, some of which are finite differences and symbolic differentiation. Each of the above techniques exhibit inefficiencies that prove expensive with respect time or space. However, Automatic Differentation(AD) attempts to produce accurate results while costing less. It breaks down each function valuation into a sequence of elementary arithmetic operations and then applies the chain rule of calculus to each of the individual operations.

Considering the compiler executes complex functions by breaking them into a partially ordered sequence of elementary functions, AD harnesses a similar technique to compute derivatives. The elementary functions typically have known derivative functions. While performing differentiation on forward-mode, it differentiates the function while evaluating it by applying the chain rule. Alternatively, it saves the entire computational tape and computes derivatives in reverse order for reverse-mode AD. In theory, one requires a well-structured file that computes the value of the objective function and an AD package computes first or second order derivatives as required. For the purpose of this project ADMAT 2.0[11] was used for computing gradients, Jacobians and Hessians.

The gradient: Consider a differentiable function $f(x) : \mathbb{R}^n \to \mathbb{R}$. It turns out that reverse-mode AD computes the gradient in time proportional to the time taken to evaluate the function itself i.e, proportional to $\omega(f)$. This is remarkable compared to the time

clocked by finite differencing $n.\omega(f)$. The absence of size n factor from the computational time brings about a considerable difference, however in order to access the variables in reverse order all the intermediary variables demand to be saved. This space utilization is not a major downside as it can be mitigated by adopting a structured approach.

The Jacobian: Consider a vector-valued function $F(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{bmatrix}$ where for each i, $f_i : \mathbb{R}^n \to \mathbb{R}^i$ and f_i is differentiable. AD can be used to evaluate the Jacobian J in time proportional to $min(m, n).\omega(F)$. The running time can further be reduced to $\chi(J).\omega(F)$ where $\chi(J)$ is a measure of the sparsity in J. It exploits the sparsity structure of the Jacobian matrix by employing a graph coloring technique.

The Hessian: The usual and more accurate technique to compute the Hessian is to simply compute second-order derivatives by supplying the code that evaluates the function. There exists an alternative technique that is quicker which involves computing the gradient term and then applying finite differences to compute the second-order derivatives. This alternative approach is particularly useful when one anticipates a sparse Hessian, reducing the running time to $\chi(H).\omega(f)$ from $n.\omega(f)$, where $\chi(H)$ is the chromatic number of the adjacency graph of H [7].

A.2 Computing Greeks

In mathematical finance, the Greeks are quantities representing the sensitivity of derivative-prices to change in underlying parameters on which the value of an instrument or portfolio of financial instruments is dependent. The name is used because the most common of these sensitivities are denoted by Greek letters (as are some other finance measures). Financial instruments such as $\Delta, \Gamma, \rho, \Theta$ and ν are used to model the behavior of options.

Each Greek measures a different dimension to the risk in an option position. Namely, the underlying risky asset S ,the time to maturity $\tau = T - t$ or r,the interest rate or σ the volatility.

Mathematically, this is quantified as the rate at which the value of the instrument V changes with respect these parameters.

$$\Delta = \frac{\partial V}{\partial S}, \quad \Gamma = \frac{\partial^2 V}{\partial S^2}, \quad \rho = \frac{\partial V}{\partial r}, \quad \Theta = \frac{\partial V}{\partial \tau} \quad \nu = \frac{\partial V}{\partial \sigma}.$$

The Black-Scholes-Merton formulas [3] for pricing European call and put options are as follows:

$$C(S,0) = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2)$$
$$P(S,0) = Ke^{-rT}\Phi(-d_2) - S_0\Phi(-d_1)$$

where

$$d_1 = \frac{ln(\frac{S_0}{K}) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$
$$d_2 = \frac{ln(\frac{S_0}{K}) + (r - \sigma^2/2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}$$

The function $\Phi(x)$ is the cumulative probability distribution function for a standardized normal distribution.

We use the approximation to $\Phi(x)$ given by Abramowitz and Stegun [1]

$$erf(x) \approx 1 - (a_1 t + a_2 t^2 + \cdots + a_5 t^5)e^{-x^2}, \quad t = \frac{1}{1 + px}$$

(maximum error: 1.5107) where $x \geq 0$

p	0.3275911
a_1	0.254829592
a_2	0.284496736
a_3	1.421413741
a_4	1.453152027
a_5	1.061405429

Though there exist closed form expression for the Greeks, we use the 'feval' function in ADMAT 2.0 to compute derivatives. This helps in improved accuracy for the required Jacobian, gradient and Hessian.

A.3 A Note on the Smoothed Counting Function

In Section 3.2, we defined the smoothed approximation to the counting function:
For $\Delta \neq 0$,

$$TC_\Delta(x) = \begin{cases} 1 & x \leq -\Delta - \varepsilon \\ 1 - \frac{m}{4\varepsilon}(x + \Delta + \varepsilon)^2 & -\Delta - \varepsilon < x \leq -\Delta + \varepsilon \\ \frac{1-m\varepsilon}{(\Delta-\varepsilon)^2}x^2 & -\Delta + \varepsilon < x \leq \Delta - \varepsilon \\ 1 - \frac{m}{4\varepsilon}(x - \Delta - \varepsilon)^2 & \Delta - \varepsilon < x \leq \Delta + \varepsilon \\ 1 & x > \Delta + \varepsilon \end{cases} \tag{A.1}$$

where $\varepsilon = \frac{\Delta}{10}$ and $m = \frac{20}{19\Delta}$. This function is symmetric with respect to x and has the following properties:

$$\begin{aligned}
\lim_{x\to(-\Delta-\varepsilon)^-} TC_\Delta(x) &= \lim_{x\to(-\Delta-\varepsilon)^+} TC_\Delta(x) = 1 \\
\lim_{x\to(-\Delta+\varepsilon)^-} TC_\Delta(x) &= \lim_{x\to(-\Delta+\varepsilon)^+} TC_\Delta(x) = 1 - m\varepsilon \\
\lim_{x\to(\Delta-\varepsilon)^-} TC_\Delta(x) &= \lim_{x\to(\Delta-\varepsilon)^+} TC_\Delta(x) = 1 - m\varepsilon \\
\lim_{x\to(\Delta+\varepsilon)^-} TC_\Delta(x) &= \lim_{x\to(\Delta+\varepsilon)^+} TC_\Delta(x) = 1 \\
\lim_{x\to(-\Delta-\varepsilon)^-} TC'_\Delta(x) &= \lim_{x\to(-\Delta-\varepsilon)^+} TC'_\Delta(x) = 0 \\
\lim_{x\to(-\Delta+\varepsilon)^-} TC'_\Delta(x) &= \lim_{x\to(-\Delta+\varepsilon)^+} TC'_\Delta(x) = -m \\
\lim_{x\to(\Delta-\varepsilon)^-} TC'_\Delta(x) &= \lim_{x\to(\Delta-\varepsilon)^+} TC'_\Delta(x) = m \\
\lim_{x\to(\Delta+\varepsilon)^-} TC'_\Delta(x) &= \lim_{x\to(\Delta+\varepsilon)^+} TC'_\Delta(x) = 0
\end{aligned}$$

where TC'_Δ refers to the first derivative of TC_Δ. From the definition of nonconvexity,

$$1 = TC_\Delta\left(\frac{10\Delta + 0}{2}\right) > \frac{TC_\Delta(10\Delta)}{2} + \frac{TC_\Delta(0)}{2} = \frac{1}{2} + 0$$

Thus we can see that TC_Δ is a continuously differentiable piecewise quadratic and nonconvex function.

A.4 Global Optimization Strategy

Finding an arbitrary local optimum is relatively straightforward by using classical local optimization methods. However, finding the global minimum (or maximum) of a function is far more challenging.

All the global optimization solvers in MATLAB [12] have a trade off between probability of finding the optimizer versus running time. GlobalSearch is befitting to the requirements of Problem (3.5). The problem can be written generally as:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & h(\mathbf{x}) = \mathbf{0} \\ & g(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{lb} \leq \mathbf{x} \leq \mathbf{ub} \end{aligned} \tag{A.2}$$

where f is the objective function, g is the set of inequality constraints, h is the set of equality constraints, lb is the lower bound on the variable and ub is the upper bound on the variable. The bound constraints can also be incorporated into g. One can write out the first order optimality conditions and the Karush-Kuhn-Tucker conditions [4] for the problem.

The global solver requires an algorithm to find the local minimum of a constrained nonlinear problem. A gradient-based interior point method [5, 6, 14] called fmincon in MATLAB is employed for this purpose. To solve a constrained nonlinear problem of the following form,

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & h(\mathbf{x}) = \mathbf{0} \\ & g(\mathbf{x}) \leq \mathbf{0} \end{aligned} \tag{A.3}$$

The algorithm works by solving a series of approximate local minimization problems. These are created by adding a *barrier* term to the objective. For any $\mu > 0$,

$$\begin{aligned} \min_{\mathbf{x},\mathbf{s}} \quad & f_\mu(\mathbf{x},\mathbf{s}) = f(\mathbf{x}) - \underbrace{\mu \sum_i ln(s_i)}_{\text{barrier term}} \\ \text{subject to} \quad & h(\mathbf{x}) = \mathbf{0} \\ & g(\mathbf{x}) + \mathbf{s} = \mathbf{0}. \end{aligned} \tag{A.4}$$

The dimension of the slack variable vector $\mathbf{s}$ is the same as number of inequality constraints in g. The slack variables are positive, to ensure $ln(s_i)$ is finite. It is trivial to see that for $\mu \to 0$, the minimizers of Problem A.3 and Problem A.4 coincide.

Notice that Problem A.4 is an equality constrained nonlinear problem and is thus easier to solve.

Algorithm 2 FMINCON- Finding a local minimizer of a constrained nonlinear problem

1: **Input**: Starting point $\mathbf{x}_0$ in $\mathbb{R}^n$ and the known quantities in Problem A.3.
2: **Output**: A local minimizer $\mathbf{x}_*$ to Problem A.3.
3: **while** Exit condition satisfied **do**
4: At each iteration, solve the approximate minimization problem A.4 by taking one of the two steps described below.
5: *Direct/Newton Step*- Solve the KKT equations for Problem A.4 via linear approximations. If this step fails, move to the next step.
6: *Conjugate Gradient Step*- If the direct step fails, then a trust region based approach is used in the CG step.
7: Evaluate the *merit* function $f_\mu(\mathbf{x}, \mathbf{s}) + \nu \big\| (h(\mathbf{x}), g(\mathbf{x}) + s) \big\|$. The parameter ν increases in higher iterations.
8: If an attempted step does not decrease the merit function by a certain threshold, the current step is marked as failed. The step also fails if either the objective function or a nonlinear constraint becomes Inf or NaN.
9: If the attempted step succeeds, update the iterate. Else, try a shorter step.
10: **end while**

The exit conditions are usually certain tolerances or thresholds which can be user-selected. The algorithm also produces certain *exitflags* which describe whether the solution is optimal or now. If the problem were infeasible, the output is the minimum of the maximum constraint value. Note that this method requires the objective function and the constraints to be continuous.

The algorithm for global solver uses the local solver Algorithm 2 at multiple starting points which are generated using a scatter search technique.

Algorithm 3 GlobalSearch-Finding the global minimizer given the starting point

1: **Input**: Starting point $\mathbf{x}_0$ in $\mathbb{R}^n$ and the known quantities of Problem A.2.
2: **Output**: The global optimizer $\mathbf{x}_*$ of Problem A.2.
3: Run the local solver Algorithm 2 from $\mathbf{x}_0$ and obtain the output $\mathbf{x}^*$.
4: If there is a feasible solution, find the radius of *basin of attraction*. Basin of attraction is the set of initial values which outputs the same local minimum upon applying steepest descent. In GlobalSearch algorithm, these are spherical.
5: At the initial point, calculate the *score* function, which is the weighted sum of objective function and its constraint violations. The initial weight for constraint violations is 1000 and this is updated through the algorithm.
6: Generate *trial points*, which are a set of potential starting points. Use a scatter search algorithm and find a number of trial points (say 1000) which respect the bound constraints in Problem A.2.
7: Select a subset of trial points and choose the point $\mathbf{t}$ with the highest score function. Delete the subset from the list of trial points. Run Algorithm 2 for $\mathbf{t}$ and obtain the output $\mathbf{t}^*$.
8: Initialize Basins, Counters, Threshold as follows:

- **Threshold:** Set `localSolverThreshold` to be the lower of objective function values at $\mathbf{x}_0^*$ and $\mathbf{t}^*$. If these point do not exist or are infeasible, set `localSolverThreshold` to be the penalty function at $\mathbf{t}$.
- **Basins:** Set the basin for initial point `Basin_x0` to be $B(\mathbf{x}_0^*, \|\mathbf{x}_0 - \mathbf{x}_0^*\|)$. Set the basin for trial point `Basin_t` to be $B(\mathbf{t}^*, \|\mathbf{t} - \mathbf{t}^*\|)$. The basins may overlap.
- **Counters:** There are two counters. Each counter is the number of consecutive trial points that lie within the corresponding basin and have score function greater than `localSolverThreshold`. Initialize the counters to 0.

9: **Main loop:** Iteratively examine trial points to see if Algorithm 2 runs. This happens if the trial point under consideration $\mathbf{p}$ does not lie in any basin and score function at $\mathbf{p}$ is less than `localSolverThreshold`.
10: When Algorithm 2 runs for $\mathbf{p}$, let the solution be $\mathbf{p}^*$. Update the basin radius to $\text{dist}(\mathbf{p}, \mathbf{p}^*)$ and the threshold to score function value at $\mathbf{p}$.
11: When Algorithm 2 does not run for $\mathbf{p}$, increment the counter of every basin containing $\mathbf{p}$ and set the rest to zero. Multiply the radii of the basins and penalize the threshold.
12: **Exit condition:** GlobalSearch terminates if we run out of trial points or if maximum time limit is reached.
13: Create `GlobalOptimSolution` using the successful runs of Algorithm 2 and order from best to worst objective values. `GlobalOptimSolution` contains all the good solution points and their objective values. The best output is set to $\mathbf{x}^*$ and algorithm terminates.

All the global solvers used in the Global Optimization [12] framework take advantage of multicore processors by activating parallelization to speed up the computation. Though GLOBALSEARCH does not have the ability to distribute and solve starting points using multiple cores, the estimation of gradients for FMINCON are executed in parallel.

References

[1] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, volume 55. Courier Corporation, 1964.

[2] Dimitris Bertsimas, Christopher Darnell, and Robert Soucy. Portfolio construction through mixed-integer programming at grantham, mayo, van otterloo and company. *Interfaces*, 29(1):49–66, 1999.

[3] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654, 1973.

[4] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[5] Richard H Byrd, Jean Charles Gilbert, and Jorge Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89(1):149–185, 2000.

[6] Richard H Byrd, Mary E Hribar, and Jorge Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.

[7] Thomas F. Coleman. Lecture notes in portfolio optimization models, 2016.

[8] Thomas F Coleman, Yuying Li, and Jay Henniger. Minimizing tracking error while restricting the number of assets. *The Journal of Risk*, 8(4):33, 2006.

[9] Thomas F Coleman and Wei Xu. *Automatic Differentiation in MATLAB using ADMAT with Applications*. SIAM, 2016.

[10] John C Hull. *Options, futures, and other derivatives*. Pearson Education India, 2006.

[11] Cayuga Research Associates LLC. Admat 2.0. http://www.cayugaresearch.com/.

[12] MATLAB. Global optimization toolbox release 2016b.

[13] MATLAB. *Version 9.1.0 (R2016b)*. The MathWorks Inc., Natick, Massachusetts, 2016.

[14] Richard A Waltz, José Luis Morales, Jorge Nocedal, and Dominique Orban. An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Mathematical programming*, 107(3):391–408, 2006.

[15] Su Yan. A graduated smoothing method for the portfolio allocation problem with transaction costs, 2016.

[16] Yinyu Ye. Interior point algorithms-theory and analysis., 1998.