FM50 - Pricing and hedging volatility index options

Teemu Pennanen Spyridon Pougkakiotis

This document describes one of the available topics for the MSc project in Financial Mathematics. The topic here is static portfolio optimization and option pricing in an incomplete market setting. Optimal portfolios and option prices are computed numerically by employing integration quadratures and interior point solvers for convex optimization.

The project consists of three parts. The first part provides a review of the relevant theories and techniques while the second part implements indifference pricing for certain exotic options on the VIX index. In the third part, the student is free to develop the topic further in a direction of their choice.

Part 1: Mathematical and financial background

In the first part of the thesis, the student should describe the relevant techniques and theories that are used in the later parts of the thesis. This part should also provide a discussion of the relevant literature with appropriate references.

Classical references on indifference pricing include [4] on continuous time financial models and [2] on actuarial premium calculations. More recent references can be found in the collection [3]. Further references and extensions to illiquid market models can be found in [5].

Studying books and published journal articles is an important part of the project. Various sources are available also on the internet (e.g. `arxiv.org`, `ssrn.com`, `repec.org`,...) but one should pay attention to the validity of results found in work that has not been peer-reviewed.

Part 2: Options Portfolio Optimisation and Valuation

This part starts by developing a portfolio optimization model for VIX options. The model is then used to compute indifference prices for European-style options written on the VIX index. The hedging instruments will consist of cash as well as call and put options on VIX, all with maturity on the 17/07/2024. The option specifications and quotes to be used in the computations below are given in a separate CSV file VIX_Options_Quotes.csv. They were obtained at 3.45pm

on the 21/06/2024, with a starting value for VIX at 13.20. Bid- and ask-prices are per one option and quoted in dollars. The file also gives the numbers of options available at the bid- and ask-prices. The risk-free interest rate for both lending and borrowing cash is assumed to be zero. For simplicity, we assume that there are no limits on lending or borrowing cash.

A simple VIX model

Following [1], we assume that the daily values of the logarithm ξ of the VIX index follow a discrete-time Ornstein-Uhlembeck process of the form

$$\Delta\xi_t = -a(\xi_{t-1} - \bar{\xi}) + \sigma\epsilon_t \quad t = 0, 1, \ldots \tag{1}$$

where $\Delta\xi_t := \xi_t - \xi_{t-1}$ and ϵ_t are i.i.d. Gaussian variables with zero mean and unit variance. The file VIX-daily-opening-prices.csv contains daily opening values of VIX from 01/02/1990 to 11/04/2024.

Task 1. *Using least squares regression, estimate the parameters of model* (1) *and give their values in a table of the form:*

a	$\bar{\xi}$	σ

Table 1: Estimated parameters of Model (1)

Task 2. *Derive mathematical expressions for the mean and standard deviation of ξ_T at the options maturity date and give their numerical values in a table of the form:*

$E[\xi_T]$	$\sigma(\xi_T)$

Table 2: Mean and standard deviation of the logarithm of the VIX at maturity

Based on these values, calculate the median of the VIX index at maturity.

Portfolio optimisation and indifference pricing

Indifference prices will be defined in terms of a portfolio optimization with financial liabilities. In this study, we will consider *static hedging* where the hedging portfolio is chosen once at the beginning and held constant until liquidation at maturity. The corresponding hedging problem can be written as

$$\begin{aligned} &\text{minimize} \qquad \mathcal{V}\left(c(\xi_T) - \sum_{j\in J} P^j(\xi_T)x^j\right) \quad \text{over} \quad x \in D \\ &\text{subject to} \qquad \sum_{j\in J} S^j(x^j) \le w, \end{aligned} \tag{P}$$

where c is a random claim to be hedged, w is a given initial wealth, J is the set of traded assets (with the convention that $0 \in J$ corresponds to the risk-free asset), $S^j(x^j)$ is the cost of buying x^j units of asset j at time $t = 0$, $D \subseteq \mathbb{R}^J$ is the set of feasible portfolios, $P^j(\xi_T)$ is the payout of one unit of asset $j \in J$ at time $t = T$ and $\mathcal{V}$ is a nondecreasing convex function on the space of random cash-flows at time $t = T$.

The objective function in (P) is defined in terms of a function $\mathcal{V}$ on the space L^0 of real-valued random variables. The function $\mathcal{V}$ describes the investor's preferences over uncertain net expenditure at time $t = T$. In this study, we take

$$\mathcal{V}(c) := \frac{1}{\lambda} \ln E \exp(\lambda c/\hat{w}), \tag{2}$$

where $\lambda > 0$ is a given "risk aversion" parameter and E denotes the expectation with respect to the underlying probability measure to be specified below. Throughout this project, we set $\hat{w} := 100,000$. This constant simply specifies the units of accounting in the sense that wealth is measured in multiples of $100,000$USD. The set of feasible portfolios is given by

$$D := \prod_{j \in J} [-q_b^j, q_a^j]$$

where q_b^j and q_a^j are the quantities available at the best bid- and ask-prices.

We will assume that the claim c only depends on the value of the VIX index at time $t = T$. It follows that all the random quantities in the above problem are uniquely determined by VIX at maturity.

Task 3. *Use the quadrature described in the appendix to approximate the expectation in the objective function by a finite sum of convex functions of the portfolio. Write down the resulting optimization problem in terms of a finite set of linear inequality constraints and verify that the problem is convex.*

Task 4. *Assume that the agent has initial position of $(w, c) = (10^5, 0)$. Using an interior point solver (see the appendix), find the optimal portfolio for the risk aversion parameters $\lambda = 2$ and $\lambda = 20$, respectively. In both cases, visualize the optimal portfolio, plot the kernel density estimate of the terminal wealth distribution as well as the terminal wealth as a function of the VIX at maturity.*

Indifference pricing

We will denote the optimum value of (P) by

$$\varphi(w, c) := \inf_{x \in D} \left\{ \mathcal{V}\left(c(\xi_T) - \sum_{j \in J} P^j(\xi_T) x^j \right) \;\middle|\; \sum_{j \in J} S^j(x^j) \leq w \right\}.$$

For an agent with financial position described by $\bar{w}$ units of cash at time $t = 0$ and a liability of delivering a random cash-flow $\bar{c}$ at time $t = T$, the *indifference*

price for selling a claim c is defined by

$$\pi_s(\bar{w}, \bar{c}; c) := \inf\{w \in \mathbb{R} \,|\, \varphi(\bar{w} + w, \bar{c} + c) \leq \varphi(\bar{w}, \bar{c})\}.$$

This is the least price at which the agent could sell the claim c without worsening her financial position as measured by the optimum value of (P). Analogously, the indifference price for buying c is given by

$$\pi_b(\bar{w}, \bar{c}; c) := \sup\{w \,|\, \varphi(\bar{w} - w, \bar{c} - c) \leq \varphi(\bar{w}, \bar{c})\}.$$

The indifference prices can be approximated numerically by a line search algorithm with respect to w and solving problem (P) by approximation of the expectation and solving the resulting optimization problem by using available optimization packages.

Task 5. *Given the portfolio optimization problem* (P) *specified earlier, show that the indifference price for selling and buying prices can be expressed as*

$$\pi_s(\bar{w}, \bar{c}; c) = \hat{w}\left(\varphi(0, \bar{c} + c) - \varphi(0, \bar{c})\right)$$

and

$$\pi_b(\bar{w}, \bar{c}; c) = \hat{w}\left(\varphi(0, \bar{c}) - \varphi(0, \bar{c} - c)\right).$$

Task 6. *Consider an investor with initial position* $(\bar{w}, \bar{c}) = (10^5, 0)$ *and risk aversion* $\lambda = 20$ *and compute the indifference buying and selling prices for one unit of a digital option with strikes* $K = 5k$*, for* $k = 0, 1, \ldots, 10$*, and plot the prices as functions of the strike. We assume that a digital call option pays* $1,000$ *USD if the value of VIX at maturity is higher than the strike* K*. Otherwise, the option payout is zero.*

Part 3

In this part of the thesis, the student is invited to engage in independent research by developing the above topics in a direction that could be practically relevant and useful or otherwise interesting. Possible topics could include:

- alternative models for the underlying risk factors;
- different choices of integration quadratures;
- alternative descriptions of risk preferences;
- margin requirements, different lending and borrowing rates;
- dynamic trading strategies;
- duality and optimality conditions;
- ...

Appendix

Approximating the objective by an integration quadrature

Assume that the distribution of the logarithmic VIX at maturity has probability density function p_{ξ_T} on the real line, with expected value denoted by μ and standard deviation denoted by σ. When the function $\mathcal{V} : L^0 \to \mathbb{R}$ is given by (2), the objective function of the portfolio optimization problem can be rewritten as

$$\mathcal{V}\left(c(\xi_T) - \sum_{j\in J} P^j(\xi_T)x^j\right) = \frac{1}{\lambda}\ln\left(\int_{-\infty}^{\infty} f(x,\xi)p_{\xi_T}(\xi)d\xi\right),$$

where

$$f(x,\xi) := \exp\left(\frac{\lambda}{\hat{w}}\left(c(\xi) - \sum_{j\in J} P^j(\xi)x^j\right)\right).$$

One can approximate the integral above by a quadrature of the form

$$\int_{-\infty}^{\infty} f(x,\xi)p_{\xi_T}(\xi)d\xi \approx \sum_{i=1}^{M} f(x,\xi_i)p_{\xi_T}(\xi_i)\Delta\xi_i$$

where ξ_i are chosen quadrature points and $\Delta\xi_i = \xi_i - \xi_{i-1}$. In the computations, use the equidistant quadrature points $\{\xi_i\}_{i=0}^{M}$, with $M = 100$, where $\xi_0 = \mu - 9\sigma$ and $\xi_M = \mu + 9\sigma$ (since the points are equidistant, we have that $\Delta\xi_i = 18\sigma/M$).

Reformulation of the budget constraint

Let any $j \in J \setminus \{0\}$. The cost functions S^j are given by

$$S^j(x^j) = \begin{cases} p_a^j x^j & \text{if } x^j \geq 0, \\ p_b^j x^j & \text{if } x^j \leq 0, \end{cases}$$

where p_b^j and p_a^j are the bid and ask prices of asset $j \in J\setminus\{0\}$ (where we assume that 0 corresponds to the risk-free asset). Such nondifferentiable functions are not supported by common interior point solvers. However, we can express the trading cost as

$$S^j(x^j) = p_a^j x_+^j - p_b^j x_-^j,$$

where x_+^j and x_-^j are the positive and negative parts of x^j. We can thus express the original budget constraint with the system of linear constraints

$$\begin{cases} \sum_{j\in J\setminus\{0\}}(x_+^j p_a^j - x_-^j p_b^j) + x^0 \leq w, \\ x^j = x_+^j - x_-^j, \quad x_+^j \geq 0, \quad x_-^j \geq 0, \quad \forall j \in J \setminus \{0\}. \end{cases}$$

Numerical solution of problem (P) using an interior point solver

In order to perform the numerical optimization of the models arising in this project, we recommend using the *MOSEK* interior point solver. MOSEK is a commercial optimization software that is available freely via an academic license. In Python, MOSEK can be run using the `cvxpy` package. In what follows, we explain how to obtain an academic license for MOSEK, how to load it into your preferred Python environment (specifically, CoCalc or Google Colab), as well as how to install MOSEK and `cvxpy` on your Python Jupyter (CoCalc or Google Colab) session. Then, we will discuss how to upload and load the problem data into your Jupyter session (for both CoCalc and Google Colab).

Disclaimer: You may use either CoCalc or Google Colab. If you decide to use Google Colab, you should be aware that KCL does not have institutional support in case unexpected things happen.

MOSEK Personal Academic License

In order to use MOSEK, you first need to obtain an *academic license*. To that end, you need to visit

`https://www.mosek.com/products/academic-licenses/`

and press the button saying "**Request Personal Academic License**". Then, you need to include your details (making sure to **give MOSEK your KCL email** and not your personal one). You are then expected to receive a file called "Mosek.lic". Download this file and store it anywhere in your computer.

Instructions for programming in CoCalc

Using MOSEK in CoCalc Login in `cocalc.com` and create a new folder named ".mosek" where you will upload the Mosek license. On the left side in CoCalc, click on New, select folder, and call it ".mosek/". This is depicted in Figure 1.

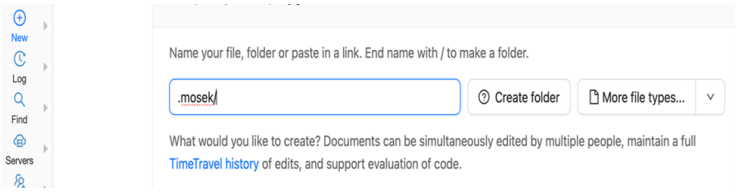

Figure 1: Create a folder .mosek to store the MOSEK license file (CoCalc MOSEK installation)

Once you have created this folder, you need to upload the license file (that you obtained from MOSEK) and place it in that newly created .mosek folder

(e.g., by drag and drop). An example of how this can be done is depicted in Figure 2.

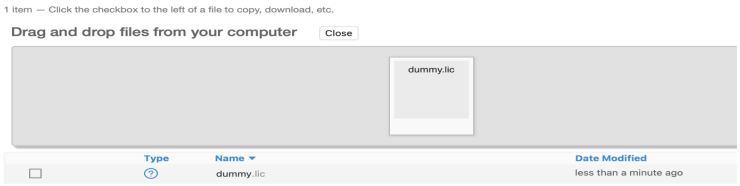

Figure 2: Upload the MOSEK license file in the .mosek folder (CoCalc MOSEK installation)

By clicking "Explore" on the left you will be taken back to your folder. You can now write code and call MOSEK with your license, using the following code snippet at the top of your Jupyter notebook:

```
import os
# In the command below we break the line for visibility.
# You do not need to do that in  your code
os.environ["MOSEKLM_LICENSE_FILE"] = \
    "/home/user/.mosek/mosek.lic"

!pip install mosek
%pip install cvxpy[MOSEK]
```

Note that `cvxpy` is the package you need to use to call MOSEK (which should, in turn, be used to solve the optimization problems associated with this project).

How to upload and use the VIX dataset in CoCalc There are several ways to upload and use a ".csv" dataset in CoCalc. Perhaps the easiest way involves creating a folder within your project and storing the data file in that folder. Then, you can directly import the data within your Jupyter notebook. This can be done using the following code:

```
import pandas as pd
# Below, insert the proper location of the file
dataset_path='./data_file/file.csv'
vix_data = pd.read_csv(dataset_path)
```

In order to check whether the file has been imported properly, run the following code:

```
# Extract relevant columns
strikes = vix_data['strike'].tolist()
option_types = vix_data['option_type'].tolist()
bid_sizes = vix_data['bid_size_1545'].tolist()
bid_prices = vix_data['bid_1545'].tolist()
```

```
ask_sizes = vix_data['ask_size_1545'].tolist()
ask_prices = vix_data['ask_1545'].tolist()

# Print the first few values for verification
print("Strike Prices:", strikes[:5])
print("Option Types:", option_types[:5])
print("Bid Sizes:", bid_sizes[:5])
print("Bid Prices:", bid_prices[:5])
print("Ask Sizes:", ask_sizes[:5])
print("Ask Prices:", ask_prices[:5])
```

Instructions for programming in Google Colab

In order to use Google Colab, you need to create a **gmail** account (if you do not already have one) and use it to sign in to Google Colab using the following link: `https://colab.research.google.com/`.

Using MOSEK in Google Colab. Create a Jupyter Notebook on Google Colab. At the top of the notebook, include the following code:

```
!pip install mosek
%pip install cvxpy[MOSEK]
import cvxpy as cp
```

Again, note that `cvxpy` is the package you need to use to call MOSEK. Then, in order to properly include the MOSEK license, add the following code immediately after:

```
# Upload the MOSEK license
from google.colab import files
uploaded = files.upload()
import os
# Create the directory if it doesn't exist
os.makedirs('/root/mosek', exist_ok=True)
# Move the license file to the correct location
!mv mosek.lic /root/mosek/mosek.lic
```

After running this code cell, you will be asked to upload the license file. Locate the file "Mosek.lic" in your computer and upload it on Google Colab.

How to upload and use the VIX dataset in Google Colab There are several ways to upload and use a ".csv" dataset in Google Colab. Perhaps the easiest way involves storing the data file in your Google Drive, and mounting your drive to Google Colab. Then, you can locate the file from your drive and directly import it to the project. This can be done using the following code:

```
import pandas as pd
from google.colab import drive
drive.mount('/content/drive')
# Below, insert the proper location of the file
dataset_path='/content/drive/My Drive/file.csv'
vix_data = pd.read_csv(dataset_path)
```

In order to check whether the file has been imported properly, run the following code:

```
# Extract relevant columns
strikes = vix_data['strike'].tolist()
option_types = vix_data['option_type'].tolist()
bid_sizes = vix_data['bid_size_1545'].tolist()
bid_prices = vix_data['bid_1545'].tolist()
ask_sizes = vix_data['ask_size_1545'].tolist()
ask_prices = vix_data['ask_1545'].tolist()

# Print the first few values for verification
print("Strike Prices:", strikes[:5])
print("Option Types:", option_types[:5])
print("Bid Sizes:", bid_sizes[:5])
print("Bid Prices:", bid_prices[:5])
print("Ask Sizes:", ask_sizes[:5])
print("Ask Prices:", ask_prices[:5])
```

How to use CVXPY

CVXPY is a Python package that allows you to set-up and solve optimization problems easily. In what follows, we explain the basic functionalities of `cvxpy`, as well as how it can be used alongside MOSEK. Further documentation on `cvxpy` can be found in `https://www.cvxpy.org/`.

The basics of CVXPY `cvxpy` allows us to model and solve optimization problems in Python. To do that, we must set up our optimization model (by specifying the problem's variables, the problem's objective function, and the problem's constraints). This can be done easily with `cvxpy` as shown below:

```
import numpy as np

# Problem data.
m = 30
n = 20
np.random.seed(1)
A = np.random.randn(m, n)
```

```
b = np.random.randn(m)

# Construct the problem.
# (variables x>=0)
x = cp.Variable(n, nonneg=True)
objective = cp.Minimize(
            cp.sum_squares(A @ x - b))
constraints = [x <= 1]
prob = cp.Problem(objective, constraints)

# The optimal objective value is returned by
#`prob.solve()` (specify MOSEK as the solver)
result = prob.solve(solver=cp.MOSEK)

# The optimal value for x is in `x.value`.
print(x.value)

# The optimal Lagrange multiplier for a
# constraint is in `constraint.dual_value`.
print(constraints[0].dual_value)
```

Below we collect some additional useful commands of `cvxpy`:

```
some_list = [1,2,3,4]
cp.log_sum_exp(cp.hstack(some_list))

# Define a numpy array
a = np.array([1, 2, 3])

# Define a cvxpy variable
x = cp.Variable(3)

# Compute the inner product using cp.multiply
inner_product = cp.sum(cp.multiply(a, x))
```

You may find additional examples showcasing different ways of using `cvxpy` in the following link: `https://www.cvxpy.org/examples/`.

Disciplined convex programming and CVXPY Let us note that the optimization problems that you need to solve belong to the so-called class of *exponential conic programming* problems, which MOSEK can readily handle. However, for MOSEK to solve the problem, one needs to write the optimization program in the appropriate (exponential conic) form. Thankfully, `cvxpy` does this automatically through a technique called *disciplined convex programming* (DCP). In order to ensure that your Python code is consistent with DCP, you need to write down the objective function of your problem in a log-sum-exp

form. To that end, you may use the `cp.log_sum_exp` of `cvxpy` to write the objective function of the discretized portfolio optimization problem. If you do this incorrectly, `cvxpy` might return a DCP error, stating that the problem is not in a DCP form.

References

[1] M. Avellaneda and A. Papanicolaou. Statistics of vix futures and applications to trading volatility exchange-traded products. *International Journal of Theoretical and Applied Finance*, 22(01):1850061, 2019.

[2] H. Bühlmann. *Mathematical methods in risk theory.* Die Grundlehren der mathematischen Wissenschaften, Band 172. Springer-Verlag, New York, 1970.

[3] R. Carmona, editor. *Indifference pricing: theory and applications.* Princeton series in financial engineering. Princeton University Press, Princeton, NJ, 2009.

[4] S. D. Hodges and A. Neuberger. Optimal replication of contingent claims under transaction costs. *Reviev of Futures Markets*, 8:222–239, 1989.

[5] T. Pennanen. Optimal investment and contingent claim valuation in illiquid markets. *Finance Stoch.*, 18(4):733–754, 2014.