Lisbon School of Economics & Management
Universidade de Lisboa

Master

Mathematical Finance

MASTER´S FINAL WORK

INTERNSHIP REPORT

Lisbon Non-Linear Trading – Equity Derivatives Exotic Trading

Henrique Silva Gameiro

October 2024

Master

Mathematical Finance

MASTER´S FINAL WORK

INTERNSHIP REPORT

Lisbon Non-Linear Trading – Equity Derivatives Exotic Trading

Henrique Silva Gameiro

SUPERVISION:

PROF.DOUTOR JOÃO JANELA

DR. RAFAEL BELO

October 2024

Acknowledgments

I would like to express my deepest gratitude to all those who have provided unwavering support during my time at ISEG.

First and foremost, I express my deepest gratitude to my supervisor, Prof. João Janela, for his guidance and advice.

I want to express my sincere appreciation to BNP Paribas and especially to the Lisbon Non-Linear Trading team for everything they taught me during my internship. I am grateful to my supervisor, Rafael Belo, for his ongoing support and guidance during my time at BNP Paribas. His mentorship has significantly influenced my professional growth.

I would like to express my deepest gratitude to my family, especially to my mom, dad, and sister, for everything they have done for me. Without their support and encouragement, I would not have been able to reach this milestone.

Lastly, I would like to extend my thanks to my friends, particularly Joana and Afonso, for their unwavering support and motivation over the years. Your encouragement has been very important to me.

Glossary

PnL - Profit and Loss

CC - Client Contributions

PDL - Prime Description Language

VIX - Cboe Volatility Index

D - Current Trading Day

D+1 – Next Trading Day

Abstract

This report outlines the main activities undertaken during my internship with the Non-Linear Trading team at BNP Paribas in Lisbon. During the internship I developed several projects using Python, one of which was an automated email that runs every day at the of the trading day, providing traders insight on Client Contributions (CC) and frictions to track the day's performance, which are then used to calculate the daily Profit and Loss (PnL). In addition, I developed an application for the Head of Equity Exotic Trading to monitor moves of certain parameters for the given underlying in a range of days given as input by the user for the corresponding book. I also contributed to the team's daily task and assisted with various analyses related to trading. This internship allowed me to deepen my knowledge in financial data analysis and automation, as well as a trading day in an investment bank.

Key Words: Phyton, PnL, CC, Frictions.

Index

Introduction

The master's program in Mathematical Finance covers a wide range of topics that can be applied across various areas within financial institutions, with one of the most exciting areas for me is trading. From the start of my academic journey, I have been particularly interested in derivatives and financial markets. Courses such as Financial Instruments and Markets, Mathematical Methods in Finance, Stochastic Calculus, and Programming Techniques played a crucial role in preparing me for the challenges of my internship at BNP Paribas, where I am currently part of the Lisbon Non-Linear Trading.

This internship, which spans seven months, has allowed me to apply my academic knowledge in a real-world context, leveraging the quantitative and programming skills I developed throughout my studies.

This report outlines the activities carried out during my internship at BNP Paribas, where I actively participated in various projects focused on developing technological solutions and automating processes. In addition to these projects, I was responsible for performing and automating daily tasks, allowing traders to focus more on the market rather than on repetitive operational activities. This automation had a direct impact on improving the team's efficiency by freeing up time and resources for more strategic tasks.

In this report, I will detail the technologies used, the challenges faced, and the knowledge gained, emphasizing the significance of these activities in my professional development.

BNP Paribas, is a leading global bank, structures its operations into three primary divisions: Corporate & Institutional Banking (CIB), Commercial Personal Banking & Services (CPBS), and Investment & Protection Services (IPS). To enhance collaboration across these business lines and better address the evolving needs of customers, employees, investors and other stakeholders, the divisions adopt a unified and integrated approach. This strategy aims to leverage each division’s expertise to ensure that customers receive continuous and comprehensive support for their long-term projects.

BNP Paribas CIB, division relies on three main areas of expertise to accompany corporate and institutional clients around the world.

- Consulting, financing and transaction banking for corporate clients and institutional investors, with Corporate Banking;
- Capital markets investment and financing, with Global Markets;
- Securities clearing, custody and services with Securities Services;

Global Markets division is responsible for market making. Market making consist in providing clients with liquidity, always providing a price whether ask or bid. The trader and the sales give the client a price based on the models used in the bank and how aggressive the bank wants them to be. The model will retrieve the fair price of the product, this price can or cannot be the real fair price but is the price that the bank believes is fair. Different banks will have different prices because the pricing models aren't the same for all banks. The aggressiveness is not the same for all clients and for all the products, some will have larger commissions (CC) however if the client is a big institution and the bank want to keep doing business with them the commissions will be reduced to keep the client close to the bank. The size has an important role in the value of the commission, for a large deal the bank takes less CC than for a small deal, however if the size of the trade in enormous the bank must increase the contributions of the client due to the bigger exposure to the market moves. When a deal is traded, hedging comes into the next step. Hedging is the technique used by market makers to mitigate risk, some risk can be mitigated by buying/selling shares (delta hedge), using futures, variance swaps, vanilla contracts and other exotic structures.

Lisbon Non-Linear Trading is a sub team within BNP Paribas's Corporate & Institutional Banking (CIB) Global Markets division. Our primary responsibilities include developing innovative technologies and services for trading, such as new applications and risk reports, as well as implementing advanced trading strategies.

2.Projects

The first project I worked on involved the development of an automated email system that runs daily at approximately 5:20 PM. This system was designed to report the PnL of CC and frictions, which are key components used in the daily PnL report, this report also shows the biggest trades of the secondary market as well as all the trades for each book. The automated email delivers a summary of these values, ensuring that the essential data is ready to be reported in the daily PnL report, which reports the performance of the team each day.

The main project I undertook during my internship was the development of a Phyton application that allows traders to analyze various financial parameters such as the at the money forward volatility, repo, smile, and curve, for a specific underlying over a specified range of days. This application also allows traders to visualize the PnL associated with each parameter.

In addition to developing new applications and reports, I also worked on implementing solutions for existing applications that occasionally encountered crashes. My task was to develop an algorithm designed to identify and filter the main roots that would cause these issues, ensuring that the application would run smoothly. This work not only improved the stability of the application but also provided a better experience for the users, enabling them to focus on their trading activities without further interruptions.

As a side project to my primary responsibilities, I initiated a project focused on developing a back-testing trading strategy centered around shorting Cboe Volatility Index (VIX) futures. This project is still in progress, as the goal is to develop several trading strategies to compare the PnL of each one. This comparative analysis will allow us to identify the most effective strategies for different market conditions.

2.1 CC and Frictions Email

In today’s dynamic financial environment, effective reporting and communication of critical information are essential for a world-leading bank such as BNP Paribas. Time is money and accurate reporting allows management to make informed decisions without needing to seek out information actively. Automated emails, which streamline communication effectively, are key to the team to ensure that the traders and management receive updates promptly.

The email of CC and Friction is one of the many automated reports within our team and the first that I have developed. There are two important concepts emerge: **Frictions** and **CC.**

Frictions – are the cost of hedging the contract in the market.

CC – refers to the additional value added to a trade or transaction by the bank, representing our commission to provide this financial service. When the sales and the trader provide the price to the client, the models calculate three key values:

- **Book Price:** This price corresponds to the price returned by the model.
- **Fair/Option Price: Book Price + Frictions**
- **Client Price: Fair/Option Price + CC**

At the end of each trading day, a PnL report is sent to Management to summarize the day’s trading performance. In this project the goal was to develop a python script to generate an automatic email with CC and Frictions.

Before proceeding with the explanation, it is essential to clarify what a PDL (Prime Description Language) is. It is the BNP Paribas proprietary file format for exchanging and storing data, it is a file that has the structure of a Python dictionary. This file is constituted by a succession of objects. Each object is defined by its “object_name” and possesses several attributes, each represented by a key/value entry in the PDL. All objects in the PDL have a mandatory attribute called “entity_kind”, of type string which describes which class the object belongs to.

Every time a deal is traded it generates a PDL with all the necessary information about the deal. To implement this project, it was necessary to find all PDLs corresponding to the trades of the day and build a pandas data frame with all the relevant information.

Having all the PDL it was necessary to parse it and find all the relevant values to this project. Using Python and digging into the PDL of each trade a data frame was built with important information such as:

- **Total CC EUR-** Corresponds to the total value of CC in EUR.

- **CC Add on EUR**- Corresponds to the liquid part of Total CC EUR. This is the value that is used to report PnL.

- **Suggested Shifts EUR -** Due to the bank's policies, I am unable to disclose this information.

- **Frictions Parameter EUR -** Due to the bank's policies, I am unable to disclose this information.

- **Frictions Model EUR -** Due to the bank's policies, I am unable to disclose this information.

- **Total Frictions = Suggested Shifts EUR + Frictions Parameter EUR + Frictions Model EUR**

Having this data, it was already possible to report the values for primary market (primary market is where new financial instruments or securities are initially issued and sold directly to investors for the first time, this market involves all the transactions up to the strike date of the contract). In the secondary market (market where existing financial instruments are traded after the strike date), the values for CC and Frictions are also key to track daily PnL. These values are calculated by a MO (middle office) team and sent to us by email. I had to parse the outlook to find the emails with the secondary trades of the day, extract the data from the email, and build a data frame to display the information.

Relevant methods applied to the python script:

- **Exchange-** In the data frame for the secondary and primary markets, there was a column with the currency of the trade**.** To have all the values in Euros it was necessary to develop a function that would retrieve the FX rate based on the currency of the trade. This function is then applied to the CC and frictions columns to convert the values to euros, ensuring consistency across the dataset.

- **Fill Book-** For the primary market, there is a column of the data frame that should contain the correct book associated with each trade. However, some trades missed this information. To address this, I developed a function that assigns the correct book for each trade based on the underlying's of the trade, ensuring that all the trades are properly categorized.

- **Groupby –** When all the strategies have a book, it is possible to group our data frame(using pandas "groupby" method) by "book" and then sum the relevant values to produce the following outcome.

Book	Total CC EUR	CC Add on EUR (1)	Suggested shifts (2)	Frictions Parameter (3)	Frictions Model (4)	**5=2+ 3 + 4**	Reserves (6)	Total (7) = 1+5 +6
SBOWLD	**150**	**133**	116	-35	918	**1282**	-12	**1270**
SCOEUR	**110**	**95**	12	10	15	**242**	-22	**220**
SBOWPB	**100**	**20**	30	10	10	**170**	-20	**150**

Figure 1 - Example of the output of the project "CC and Frictions email".

Please note, that due to privacy considerations, the values presented are fictional and do not reflect real data.

This script also shows the largest trades of the day and the biggest trades for each book; however, for the purpose of this report, I believe it is not relevant to include them.

In conclusion, it is important to point out that the values in bold will be later added on the daily PnL report to be sent to top management.

2.2- Parameter PnL Move App

Parameter PnL Move was my main project during the internship. This Python app allows traders to monitor key parameters and the correspondent PnL for a range of days. This app is designed to help traders check the positions that a given underlying has in a certain book for certain parameters.

The parameters that are studied in this app are:

- **ATMF volatility** – At-The-Money-Forward volatility refers to the expected volatility of an asset price between the current time and the given maturity of a forward contract, with the strike being the spot of a given underlying at the current time. It measures how much the price of the asset is expected to fluctuate over time.

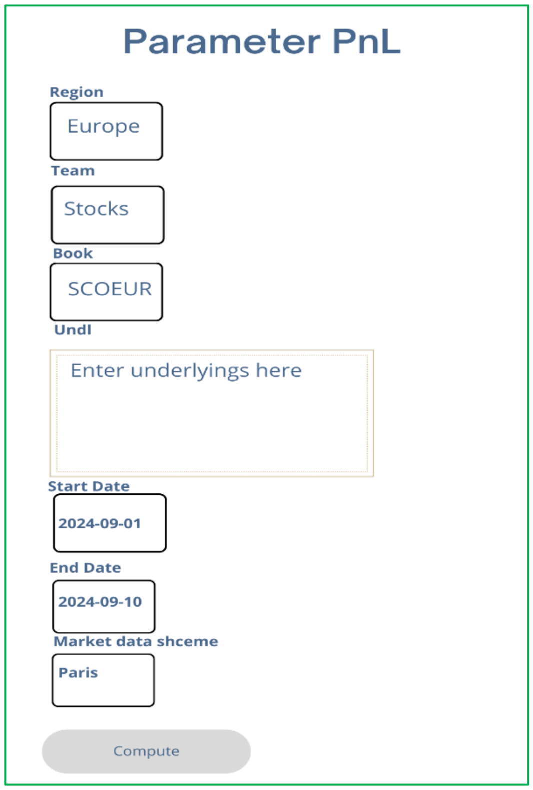
- **Repo**- Repo also known as Repurchase Agreement, can be defined as an agreement in which one party sells securities or other assets to a counterparty and simultaneously commits to repurchase the same or similar assets from that counterparty, at an agreed future date at a repurchase price equal to the origin sale plus a return on the use of the security that was initially sold.

- **Smile**- When plotting the strike and implied volatility of a group of options with the same underlying and the same expiration date it may result in a graph called volatility smile. Smile shows that options that are further in the money (ITM) or out of the money (OTM) have the highest implied volatility and that options that are at the money (ATM) have strike prices with lower volatiles. When the value of smile increases it means that the options that are on the downside (options with lower strikes) will have bigger implied volatility.

- **Curve**- Volatility curve is a parameter that studies the convexity of implied volatility (IV) across different option strike prices. When the implied volatility increases, the graph of the volatility curve will have an upward shift indicating increased market expectations for future prices.

To elucidate the development process of the app, I will explain it in three distinct sections: **Front-End**, where I built the interface of the app. **Class Equity** section will define the development of a dedicated class for equity, which includes several instance methods that have the goal to be used in pricing and other relevant processes. **Back-End** section outlines all the data manipulation to prepare data to be presented in **Front-End.**

Front-End – To start the app, I utilized the Panel library to develop the interactive and user-friendly interface. This application was developed to be deployed across multiple teams across different regions, currently used in Europe and Asia. It takes as input the region, the specific team, and all the books available for that team in that region. When a user changes the selected region, the application will dynamically update to display the teams within that region, which in turn allows the user to select the wanted team and then all the books for that same team. In addition, the app features a Text Area input widget where the users can provide the underlying to be studied (it is possible to study more than one underlying), alongside the start and end date to analyze the parameters move between each date and finally the corresponding market data scheme, this widget as three possible values Paris, New York or Hong Kong.


Parameter PnL

Region: Europe

Team: Stocks

Book: SCOEUR

Undl: Enter underlyings here

Start Date: 2024-09-01

End Date: 2024-09-10

Market data shceme: Paris

Figure 2 – Mockup of the interface of the application.

Due to privacy considerations, I am unable to present the actual interface in this report. Therefore, I have provided a simple mockup to give an idea of the app's interface and functionality.

Class Equity- Is a child class derived from an existing library used within the bank, which provides fundamental structures and functionalities for financial instruments. By extending this library class, the class Equity inherits core properties and methods, while also incorporating additional instance methods tailored to the specific needs of the project. A key attribute to this class is the attribute “instance_name” that stores the MDS code (for example DE_BMW is the MDS code for BWM) of the equity. It is important to point out that all the following methods that I will explain are to be used in future pricing.

- **get_bucket** – Is a crucial method, it plays a crucial role in obtaining market data related to the positions of the parameters that are being studied. This method will set a Bucket as an attribute of the class equity which is retrieved from an internal platform in the bank, where data is structured as a **Bucket**. In the context of this project, a **Bucket** represents the risk exposure of a given equity across multiple maturities. It encapsulates the sum of the sensitivities for all financial products in the book that contain a given equity for all the listed maturities.
  In this method, I also set as an attribute of the class equity the maturity list that comes with the **Bucket.** These maturities that come with the bucket will be used in the future for pricing Futures, Vanillas, and Dividends Swaps.

Maturity	Vega Bucket	Repo Bucket	Smile Bucket	Curve Bucket
18.Oct.2024	350	250	147	152
15.Nov.2024	1000	250	420	32
20.Dec.2024	-251	-254	452	657
21.Mar.2025	-358	52	87	2554
20.Jun.2025	2580	-30	255	569
19.Sep.2025	-145	857	652	12
19.Dec.2025	1587	658	4125	754

Figure 3 – Example of a Bucket for a given equity.

This would be an example of a bucket, for a given equity. For privacy matters, this bucket is just a representation of an actual one and the values are not accurate.

- **get_vanilla_contract-** This instance method returns a vanilla option contract as a Python object based on parameters strike and maturity this means that when calling this function, it is necessary to pass the strike and maturity as arguments of the function. This vanilla contract will be a call option with underlying equity.instance_name.

- **get_vanilla_list-** This method will retrieve a list full of vanilla call options. Basically, what this method does is iterate over the maturity list of the equity and for each maturity, it calls the function get_vanilla_contract that passes the strike and the maturity as arguments. This way a vanilla call option is created and appended to the list of vanilla call options.

- **get_future-** This method is designed to generate a Future contract for the underlying equity.instance_name. This method takes specified maturity as a parameter.

- **get_future_list-** Basically it's the same functionality as the **get_vanilla_list** function but instead it returns a list with Future contract for the given equity for all the maturities presented in the maturity list.

- **get_dividend_swap-** This method is responsible for the creation of a python object that represents a dividend swap contract for the given equity and for a given maturity that it is passed into the function as a parameter.

- **get_dividend_swap_list-** Also has the same functionality as the function **get_vanilla_list,** however it returns a list of dividend swap contracts as python objects. This function filters the maturity list and keeps only the maturities with the month of December.

This function has also other methods but for the purpose of this report I believe that they are not relevant.

Class Backend – This class will be responsible for pricing futures, call options and dividends swap contracts and for all the data manipulation. To explain this class, I will provide a detailed description of its most relevant functions.

- **populate_equity_list-** This function is one of the pillars of the project. The inputs of the underlying's will come to the backend as a list. Now, as previously reported, class equity is a child class of a class that is internal to the bank and that has its own methods. In this way, I iterate over the list of underlying's that the user inputs in the front end and I will create an instance equity using a class method from the parent class (the class that is internal to the bank), this way my instance equity will have all the market data of that given underlying (this is true because the method of the parent class will automatically load the market data for a given equity). I also added this Python object equity to a list named equity list.
  Now that we have a list with all the equities, it is necessary to iterate over the list and call the function **get_bucket.** This way our equity object will have two new attributes, the maturity list and the bucket.

- **price_futures –** Any pricing that is done is made by a server. To price this we need to send to the server an object that the server can recognize. In the bank, we use PDL as already mentioned. This PDL to be priced, needs to have a Book and a type of pricing. So, to start it is necessary to create a Book as a Python object this book will have an option list with all the options that we want to price. To fill this list, it is necessary to iterate over the equity list, in each iteration we will have an equity that has the method get_future_list that returns a list with all the futures for that given equity. Now that we have all the futures for all the equities, we will add them to the option list that is an attribute of the Python object book. To price this Book, our PDL (object that will be sent to the server that prices) needs to have one method of pricing, for privacy securities I can't disclose the types of pricing let's call it CPS, so it is necessary to create a CPS python object that will have as attribute the Book that contains all the futures that we want to price.
  After pricing the server returns a PDL with all the data about the pricing, such as premium, delta, repo literally everything. In order to get the premium of the futures, it is necessary to iterate over the pdl and to create a dictionary that will map the premiums of the futures with the maturities, these values will be used as strike when pricing a vanilla option. It is important to note that I price the futures at the end date (user input) using market data from the same date.

- **price_non_futures-** At this point, it is necessary to create the vanilla contracts and dividend swap contracts for each equity. For vanilla contracts, we iterate over our equity list, and we call the function get_vanilla_list that will retrieve a list with vanilla contracts with the strike being the premium of the futures contracts, we do the same for the dividend swap contracts. Now following the same logic as in the price futures contract function, we add these contracts to a book that is inside a Python object CPS, and we price it. This function will have as input a parameter date that will be essential to get market data for each day that the traders choose as input in the front end. For the vanilla and dividend swap contracts, the pricing is conducted twice on the end date, but with distinct sets of market data. First, the contracts are priced using market data from the start date, and then they are repriced using market data from the end date. This approach provides a clear comparison of how market variables alone impact pricing, making sure that changes in option values are due to the market data.
  The PDL that will be retrieved will have the values of the implied volatility and repo for each maturity within the maturity list.

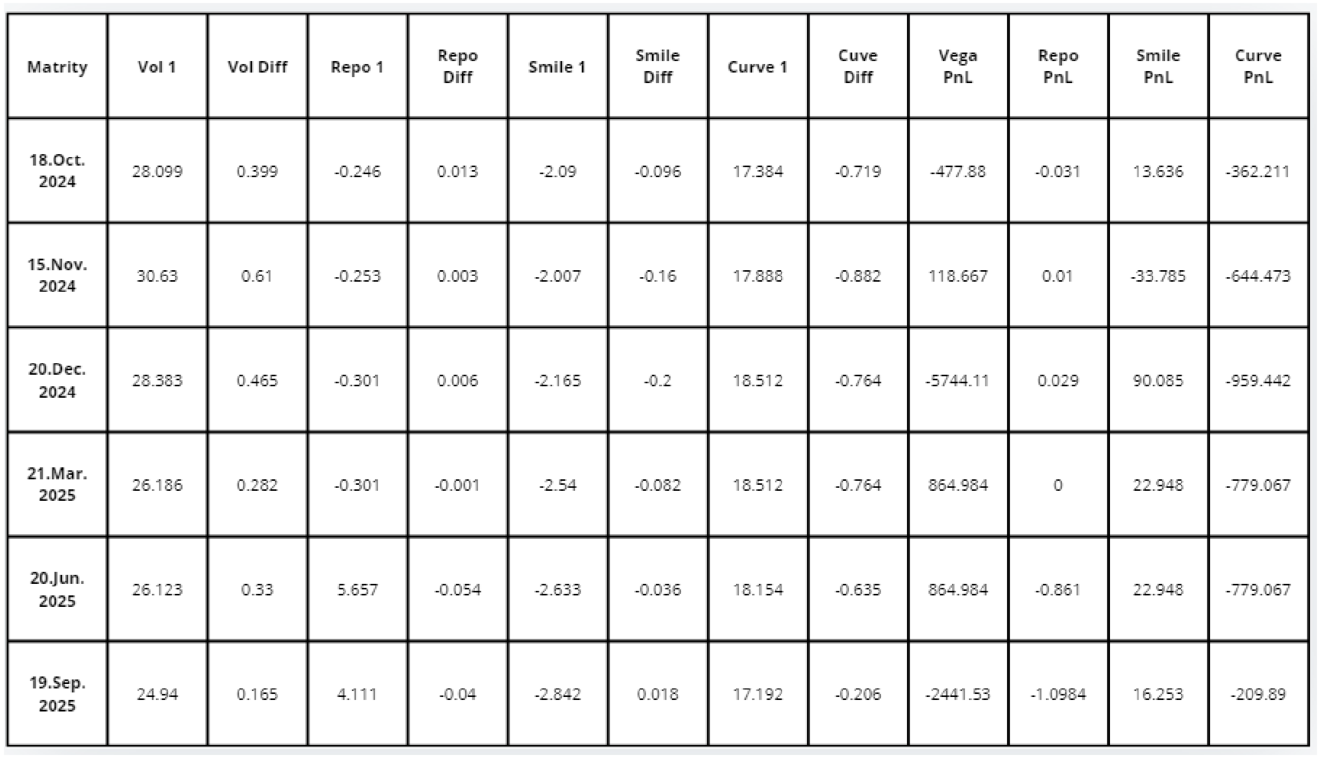
- **smile_and_curve_interpolation-** The bank also has a server that retrieves the values for volatility smile and curve for a specified date (start date, end date), however, these values correspond to a set of maturities that do not match the equity's maturity list being necessary to perform an interpolation to align the retrieved values with the specified maturities for each equity. This ensures that the volatility smile and convexity parameters are accurately adjusted to fit the equity's maturity list.

In this stage, we have all the necessary functions to construct a data frame containing relevant information.

- **Price_and_parse –** In this function, I will call all the other methods above and build a data frame for each equity. So, the first thing to do is to populate our equity list by calling the method populate_equity_list. Now we need to price our futures contracts by calling the method price_futures. Now it is necessary to call our function price_non_futures twice, once with the start date as an argument other with the end date as an argument. At this stage the only data that we don't have is smile and curve, so we call the function twice also at the start date and end date. At this point of the function, we already have all the data needed to build our data frame. In this study, I constructed a data frame where the index corresponds to the maturities of each equity, and the columns represent the parameter values for each specified date. To facilitate the analysis of movements, a "parameter diff" column was created for each parameter, which reflects the difference between the values in the “end date” column and those in the “start date” column for each respective parameter.

Having the moves for each parameter and the buckets, it is now possible to compute the PnL for each parameter by multiplying the corresponding column parameter diff by the corresponding bucket.

This approach allows a better examination of the changes in the parameters over the defined period, enabling a more detailed analysis of market dynamics. For improved clarity, I will demonstrate how it appears in the final application.



Matrity	Vol 1	Vol Diff	Repo 1	Repo Diff	Smile 1	Smile Diff	Curve 1	Cuve Diff	Vega PnL	Repo PnL	Smile PnL	Curve PnL
18.Oct. 2024	28.099	0.399	-0.246	0.013	-2.09	-0.096	17.384	-0.719	-477.88	-0.031	13.636	-362.211
15.Nov. 2024	30.63	0.61	-0.253	0.003	-2.007	-0.16	17.888	-0.882	118.667	0.01	-33.785	-644.473
20.Dec. 2024	28.383	0.465	-0.301	0.006	-2.165	-0.2	18.512	-0.764	-5744.11	0.029	90.085	-959.442
21.Mar. 2025	26.186	0.282	-0.301	-0.001	-2.54	-0.082	18.512	-0.764	864.984	0	22.948	-779.067
20.Jun. 2025	26.123	0.33	5.657	-0.054	-2.633	-0.036	18.154	-0.635	864.984	-0.861	22.948	-779.067
19.Sep. 2025	24.94	0.165	4.111	-0.04	-2.842	0.018	17.192	-0.206	-2441.53	-1.0984	16.253	-209.89

Figure 4 – Mockup of the output of the application.

It is important to note that due to privacy reasons, I cannot disclose the real layout of the application, however, this is a simple mockup with no accurate values.

This application also displays the correlation movements for certain underlying assets and zones; however, for the purposes of this report, I believe that it is not necessary to elaborate on this feature.

2.3 Physical Expiries

Although I did not develop the entire project, I contributed in such a way that it now runs automatically every day, without the need for manual execution.

This project takes into account physical expiries (delivering shares). Essentially, it involves identifying which contracts expire on the current day, for those we will dig into the PDL of each contract and verify whether the delivery will be physical (delivering shares) or in cash.

Typically, the condition to deliver shares is if the worst-performing underlying has breached a certain barrier. There are deals that are continuous monitoring, where performance is tracked daily. In these cases, we must continuously calculate the performance of the worst-performing underlying and check if it reaches the barrier. Conversely, for deals with discrete monitoring, performance is only evaluated on specific observation dates.

$$Perf(\%) = \frac{(ST - S0)}{S0} \cdot 100$$

ST –Spot of underlying asset at T.

S0 – Spot of the underlying asset at the strike date.

After identifying the deals that have breached the barrier, we need to price them and retrieve the delta value to deliver to the client. For this pricing, we will price at 95% value of the current spot price. This approach is taken into account because when a deal is close to the barrier, the variation in delta is significant due to the high gamma.

In the bank, we receive an email from MO that specifies the expiries and the number of shares to be delivered. It is important to clarify that this report is not always accurate therefore it was necessary to create this automated process.

By employing web scraping to extract information from the MO email and integrate it into our report, we aim to verify if the delta matches the one provided by MO. If there is a big discrepancy, we will notify them of a potential error on their part and ask for a recalculation.

After the pricings and the data manipulation here is an example of how the report would look alike.

Nominal	Perf	Barrier	Type	Underlying	Quantity	Counterparty	Book	Strategy
-200.000	65	70	Spot	US_TSLA	-500	X	SCOEUR	Y
-400.000	100	110	Spot	DE_BMW	-400	Z	SBOWPB	L

Figure 5 – Example of the output of the project "Physical Expiries".

In Figure 5, we can see an example of the output, where it is possible to check that at the end of the day, we would have to book 500 shares in the strategy "Y" in SCOEUR for a given counterparty "X" and 400 shares in the strategy "L" in SBOWPB for a given counterparty "L".

This report is of significant importance because prior to this implementation we had to manually check in the book all the deals expiring at the end of the day and perform the calculations to determine the value of the delta to be delivered to the client. With the implementation of this report, this process is no longer necessary, resulting in substantial time savings at the end of each day.

For internal reasons, our reports are required to run on Jupyter Hub, these reports are stored on a Bitbucket cloud, ensuring version control and collaboration among all team members. However, this task wasn't implemented in Jupyter Hub, and my task was to make sure that it could run both locally (windows) and in Jupyter Hub (Linux), this way I had to rewrite in the same Python version as the one that is in Jupyter Hub and make sure it runs in both systems.

2.4 Carry PnL

There is an application within our team called Carry PnL, this app allows traders to check the Carry PnL (the PnL due to carrying these positions over time) of the entire book.

$$Carry\ PnL = Gamma\ PnL + Implied\ Theta$$

For a basket of underlying, it is necessary to compute the implied correlation to compute the Gamma PnL. This way, based on the underlings it was needed to make a request to an interval server within the bank to ask for the implied correlation of a given basket. To have a valid response from the server it is necessary that all the underlying's within the basket are mapped in the bank correl matrix.

The problem with this application is that not all the underlying's are mapped in the correl matrix, for the ones that are not we remove them from the list of studies and run the application without any error, however when a contract is traded with one underlying that is not already in the book, it may be possible that this underlying is not mapped in the correlation matrix and if so this app will crash and it will be necessary to take time to find the underlying that is not mapped. This is a problem for our team because our goal is to have everything running well. This way it was necessary to implement an algorithm that would find which underlying's are not mapped in the correl matrix.

This algorithm follows a divide-and-conquer approach, where it attempts to retrieve a correlation matrix for a set of underlying's. If the request is unsuccessful, the algorithm systematically subdivides the set into a smaller subset to try to find the unmapped underlying's. A brief explanation follows:

1. **Initial Request for Implied Correlation:**
   - The algorithm first sends a request to the bank's server for the implied correlation matrix of all underlying's present in the book.
   - If the request is successful, it means that all the underlying's are mapped correctly, and it's possible to proceed with the calculation of the Gamma PnL.
   - If the request fails, it indicates that one or more underlying's are not mapped in the implied correlation matrix.

2. **Divide the Set of Underlying's:**
   - In case of failure, the algorithm divides the set of underlying's into two equal-sized subsets let's call it List A and List B.

- It then sends separate requests for each List to the server.
- If one subset fails while the other succeeds, the unmapped underlying(s) must be located within the List that caused the failure.

3. **Recursive Subdivision:**
   - The algorithm repeats this process recursively for the List that caused the failure, splitting it into two smaller Lists and testing each one independently.
   - This process continues until the List is reduced is reduced to maximum of 10 underlying's.
4. **One-by-One Testing:**
   - Once the List contains 10 or fewer underlying's, the algorithm begins to request the server for each underlying individually.
   - For each underlying, if the request fails, that specific underlying is identified as unmapped in the implied correlation matrix.
5. **Filtering and Exclusion:**
   - Upon the identifying the unmapped underlying(s), the algorithm excludes them from further requests

Ever since this algorithm was developed and implemented, the application has no longer crashed, significantly improving the stability and reliability of the application.

Although I haven't developed this application, I was the one who developed the algorithm.

2.5 Shorting VIX Futures

The CBOE Volatility Index, also known as VIX, is a real-time index that represents the market's expectations for the relative strength of near-term price changes of the S&P 500 index (SPX).

Investors use the VIX as a tool to assess potential risks. By understanding it, traders can gain valuable insights into market dynamics and make informed trading decisions. VIX is a way to measure market volatility. When the VIX is low, it indicates low volatility and a relatively calm market environment. Conversely, when the VIX is high, it signals high volatility.

Shorting the VIX involves selling VIX futures or VIX-related products with the expectation that the VIX will decline in value. This strategy can be profitable in a stable market environment.

I was tasked with implementing a trading strategy focused on shorting VIX futures, utilizing data spanning from 2011 to 2024. This dataset included daily spot prices of the VIX index as well as the corresponding futures contracts.

To begin with, since one of the trader's conditions was to only take short positions on days with five or more available futures contracts, I had to clean the dataset removing all the days that did not complain with this condition.

To implement this strategy, I used Python and decided to create three classes: Contract, Portfolio, and Manager.

The Contract class has attributes including the day the contract was traded, the premium paid to short the contract, the name of the contract, and the spot price at which the contract is trading, in the trading date the spot and the premium paid are the same. This spot price will be updated daily. The Contract also features a method called days_to_maturity, which returns the number of days until maturity, and a method to calculate the PnL of the contract, defined as the premium paid minus the spot price.

The Portfolio class serves as an object containing two lists of contracts: one called portfolio and another for closed trades. This class includes methods for managing contracts such as adding or removing a trade from the portfolio. It features a function that given a contract will return the number of positions shorted for that particular contract; this will help us in our strategy because one restriction will be that we don't want to have more than x positions opened for a given contract. Additionally, the Portfolio class provides a method to update the spot values of each contract within the portfolio and another method to close a trade based on specific conditions. When a trade is closed, it is removed from the portfolio list and added to the closed trades list. Furthermore, the class allows for the computation of the overall PnL of the portfolio. This is accomplished by iterating over all contracts in both lists (portfolio and closed trades) and summing the PnL of each contract.

The Manager class is designed to manage a list of portfolios, currently initialized to handle one portfolio but structured to accommodate additional portfolios in future implementations. The primary responsibility of the Manager class is to determine which contract will be shortened based on a specific strategy. Currently, the strategy involves shorting the contract with the highest premium, provided that the premium is greater than the spot price for that particular date. Additionally, there is a condition related to the existing positions in the portfolio. For example, if we intend to short a specific contract *x* and already have two open positions in that same contract, the manager will not short it once again. This way the manager will check wish contract has the second biggest premium. If this new contract's premium also exceeds the spot price, the manager will check how many positions already exist for this contract. If there are already two open positions for this new contract, the manager will refrain from shorting it as well. In this scenario, if both potential contracts do not meet the criteria, the manager will take no action.

After implementing these Python classes, the workflow begins by instantiating an empty Portfolio object, which will store both active and closed contracts, and a Manager object initialized with this portfolio to handle trade decisions. The strategy, executed through the iterate function, iterates the dataset by making decisions to short a new contract every 10 trading days. On each of these days, the Manager evaluates the potential contracts based on the conditions that the premium must exceed the spot price of the VIX and the maximum allowable position for that contract must not have exceeded (the current max is 2 contracts, but for future implementation can be different). If these criteria are met, the selected Contract is added to the portfolio. The strategy dictates that trades are only closed upon the contract maturity, at which point the matured contracts are removed from the portfolio and added to the "closed trades" list to keep track of both active and finished positions. Although the trades are only closed upon the maturity, the portfolio’s PnL is calculated and updated every single day, this is done by iterating over the portfolio and closed trades and calculate the PnL of each contract in each list this is done to track the performance of the strategy during the period of this back test.

This project, as I mentioned is, side project, that will include further implementations, such as a class strategy, that will represent the strategy that aligns with our trading objectives. This class will support multiple trading strategies, for example, it could close a trade when the PnL reaches around 20% profit, or close a trade ten days before the maturity, or follow other criteria. Each portfolio will have its own designated strategy, facilitating a more effective comparison of the PnL across different strategies.

3.Daily Task

In addition to the developed projects, I was also responsible for carrying out daily tasks, which included booking deals in the correct book, managing flows, and preparing and sending the daily PnL report. It is essential to allocate trades to the correct book/trader to ensure accurate bookings and proper deal management. Managing flows involves monitoring the inflows and outflows of each contract. Finally, preparing and sending daily PnL report is essential for informing the team and top management about the daily performance.

3.1 Booking Deals

One of my key daily responsibilities is to book deals with the correspondent book. This is a simple but crucial task that ensures that each trade is correctly allocated to the correct book. This process involves three primary books:

- **SCOEUR –** This book is the that takes trades involving both European and non-European underlying's. For example, a trade with a single non-European underlying or a basket of non-European underlying’s or either a basket of non-European, and European underlying's would be booked into SCOEUR also known as the world book.
- **SBOWPB –** This is the mono book for European underlying's, each trade only has one European underlying.
- **SBOWLD –** This book is designed for multiple European underlying’s.

Based on the underlying's present in a trade, the trades need to be allocated to their respective books.

For SCOEUR, there are some very important rules that should be taken into consideration, in the bank we have sales across the world and a lot of the deals that come to be booked come from Asia. In Asia, the salesperson uses software that executes delta to delta hedge the trade so that when the trades arrive at the book the trader does not need to delta hedge it. This way, depending on the order type of the trade, some rules must be respected.

- **Order Type**
  - **At Market –** If there is American underlying into the trade based on the order type, at market, I can only push the deal into the book when the American market is open, so only after 2:30 pm Lisbon time. This way when the deal is pushed into the book it is already delta hedge. For example, if we sell a contract, like an auto call, we will be short delta so the software will buy shares (long delta) to hedge it. If the deal is pushed in the morning the trade will not have the strike that delta will execute, and once the market opens, the software will execute delta, and the book will be long delta. This way, at the end of the day, as we are long delta, the trader will have to sell the position in order to be delta hedge. This is a huge problem because at D+1 (the next trading day) we will once again have to buy the delta because now the trade has the strike in the book (there is a MO team that will do it). In conclusion, if the stock price at the open of D+1 is bigger than the close of D (trading day) it will have a negative PnL impact.

  - **Afternoon Close –** This order type indicates that the trade will have the strike at the close of the corresponding market. If the trade has an American underlying, we can only book it at D+1 so that the deal when pushed to the book already has the value of the strike. On the other hand, if the deal has a European underlying, we need to book after the European close.

  - **Limit Order –** This order type indicates that the trade is executed once the spot reaches the value of the limit order. Only possible to book it after the spot price of the underlying reaches a certain limit order.

  - **Best effort vwap –** This order type is an algorithm called "Volume weight average price", that executes delta during the market at certain levels of spot. In this case, if the trade has an American underlying it is only possible to execute at D+1.

Although this task looks straightforward, it must be carried out with great care because if a trade is booked into the incorrect book, it will have impacts on PnL. To ensure accuracy, I was regularly in contact with traders in case of doubt.

3.2 Booking Shares for KOF

Another crucial task involves booking shares for KOFs. A KOF, or Knock-Out Forward, is a forward contract that includes a knock-out feature. This means that if the spot of the underlying reaches the value of the predetermined barrier price, the contract is automatically terminated. This is a very attractive product to investors because due to this feature a KOF contract is cheaper than a vanilla contract. Each day, I was responsible for tracking with accuracy the number of shares that the bank must deliver/receive based on the position that we take, if we are short KOF we deliver shares and if we are long in the KOF we receive shares.

Example: A contract that pays 5 shares per day at a forward price of 95 euros with a barrier price of 90 euros and the current spot of the underlying is equal to 100 euros, let's assume that this contract has 20 trading days if during this period the spot price doesn't close below the barrier price and the bank sold the KOF we will have to delivery 20*5 = 100 shares at the forward price. Otherwise, if on any trading day, the close price of the underlying closes below the barrier price knock-out event will occur, assuming that it was on the 6th trading day we will only deliver 5*5 = 25, as the knock-out day will not be considered.

The timing of this task will depend on the book. For SCOEUR it is only possible to book it at D+1 because when leaving the office, the American market hasn’t closed. For SBOWPB I can book it at the EOD as the European market is already closed.

3.3 Control Flows

Another daily responsibility involves overseeing the control of financial flows using a platform known as “Mobilis”.

Each time there is a flow MO creates a ticket, for flows involving the delivery or receipt of shares, I meticulously verify whether the shares are booked in the book with the correct direction, if the ticket says that is a delivery of shares, I should see in the book shares booked with negative direction and vice versa. If the ticket has the incorrect direction, I reject the associated ticket and inform the MO of the discrepancy. Furthermore, for flows like early terminations, expiries, or cash coupons, it is necessary to ensure that the nominal of the ticket matches the nominal reflected in the book for the relevant observation data, if it’s not the case I reject the nominal, and the ticket is sent back to MO team. Once the nominal is correct, it is necessary to confirm that the performance metrics match those recorded in the book and, if they do I proceed to accept the flow. On the other hand, if there is a mismatch, I stop the flow and request the calculations from the MO to identify the source of the error, whether it is from the book or the ticket that is incorrect. This process is simple but critical to maintain the integrity of the operations, as timely execution of these tasks is essential to avoid late future payments, if a ticket remains pending after the agreed payment date, the client may file a complaint, which would result in the bank being liable to pay a fine to the client and in this case there would be a negative impact in PnL.

In conclusion, although it looks like a simple task, it requires careful attention to detail. Each ticket must be carefully verified to ensure that the directions and nominal amounts align with what is recorded in the book. This verification must be completed before the payment date to avoid future issues with the clients. Failure to do so may result in fines and consequently, a negative impact on the bank’s PnL.

3.4 Report Daily PnL

At the end of each day, I prepare the daily PnL report in Excel, which includes all the parameters related to the PnL for each book. This report incorporates values such as CC and frictions, calculated from the “CC and Frictions” project email. Other key PnL parameters are provided by the traders, who also include a commentary on the day’s performance, hedging strategies, and their outlook for the upcoming days.

My responsibilities include running VBA macros to ensure all required data is accurately reflected in the report. Additionally, I provide a brief market commentary on the European market, for which I typically reference information from Bloomberg and Reuters. Once finalized, I distribute the report to the team, the head of trading, and senior management at BNP Paribas.

Conclusion

In conclusion, my internship with the Non-Linear Trading team in Lisbon has been an invaluable experience, allowing me to develop key skills such as Python programming and gain deeper insights into the world of trading. The projects I worked on such as the "CC Frictions Email," the "Parameter PnL" application, the algorithm implemented in Carry PnL, and “Physical Expiries” are of critical importance and are used daily by the team. These contributions have demonstrated their practical value, and due to my strong performance, I was invited to continue at BNP Paribas, furthering my journey in the industry.

References

Bennet, Colin. (2014). *Trading Volatility, Correlation, Term Structure and Skew.*

De Weert, F.J (2002). *Exotic Options Trading.*