Text

Description automatically generated with medium confidence

# QF633 PROJECT REPORT

# Group Members

PANG HONG YUN

AIDEN CHIA YANG CHERH

NG ZHIWEI

# Project Setup

We have added TestData folder to do some EDA and print our output results.

We have also implemented CMakeLists.txt based on the Makefile provided in the base path of the project for our ease of debugging.

# A screenshot of a computer Description automatically generatedStep 1

Column header read in correctly.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Ensure that the tick data of the same time frame is read in together within the same step. As seen from the log after we detected the change in timestamp, we will reset the file position to prepare for the ReadNextMsg method to process the same line again in the next step. Seen from above figure at row **541** where we ended the snap timestamp, and then we proceeded to read in the **BTC-27MAY22-44000-P** tick data.

A screenshot of a computer

Description automatically generated

Before we process the line, we can see that the position is still the same as previous figure “**97805**” and after reading line it became “**97982**”. Then we proceed to add the tick data.

A screenshot of a computer

Description automatically generated

After reading the **BTC-27MAY22-44000-P** contract we see that the timestamp changed again to **1652572801712** so we reset the position again and read the **BTC-3JUN22-35000-C** in the next step.

A screenshot of a computer

Description automatically generated

On the left is the tick data for **BTC-3JUN22-35000-C** contract. The process then repeats again, reading the same timestamp together in one step.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedWe also validated the end of the file to ensure that we will not miss the last contract’s tick data. As seen in the subset data we cut out. Seen on the left **BTC-27MAY22-30000-P** contract is being read in as well.

# Step 2

A screen shot of a computer program

Description automatically generated with low confidence

In the second step we clear the currentSurfaceRaw at line 32 when the Msg is a snapshot and then proceed to populate the currentSurfaceRaw with the tick data. And when it is just a snapshot, we will proceed to replace the currentSurfaceRaw with the updates directly.

A screen shot of a computer program

Description automatically generated with low confidence

In step 2 we also print out the timestamp, contract name and mark IV of the contract.

A screenshot of a computer

Description automatically generated

# Step 3

A screen shot of a computer program

Description automatically generated

1. Group market tick data by expiry date

As seen from line 76 above we skipped all the ITM options and use only ATM and OTM options to group them by their expired date.

1. Pass the data for each expiry date to fitsmiles function.

In line 101 we pass the data to inside the fitsmiles function then we use the smiles function to calculate the optimized fitting errors.

1. Fit the model to market data get fitting error.
   1. Take the average of BestBidIV and BestAskIV for fitting this is for the .

Our fitting error function and rest of the helper functions are implemented inside the **helper.h**.

* 1. The formula for the weights.

A black screen with colorful text

Description automatically generatedA screenshot of a computer program

Description automatically generatedAs seen from line 52 we get the max values of open interest and spread by going through the snapshot first, then we proceed to calculate the weight based on the average of the 2 factors see the following figure.

A math equations and numbers

Description automatically generatedThe CalculateFittingError function is an implementation of equation 13 in lecture material.

1. Complete the time\_listener in step3.cpp that streams the vol marks and fitting error to the output file   
   Output are inside the TestData folder.

A screenshot of a computer

Description automatically generated

1. Implement FitSmile in CubicSmile.h to return a CubicSmile that fits the tick data of a given expiry date.

# Smile Fitting Error

A screenshot of a computer program

Description automatically generated with medium confidence

As seen from line 288 – 292 we tried to make use of the last 2 points’ gradient to extrapolate the volatility, but it seems that the natural spline is a better way to go as the errors were smaller.

A screenshot of a computer

Description automatically generated with medium confidence

In the above code was our implementation of not a knot spline but this causes our error to go up too.

A screenshot of a computer program

Description automatically generated

Here is the random sampling error from both implementation and as seen from here somehow the Not-a-knot implementation gave us a higher average error after implementation, hence we decided to stick to Natural Spline implementation.

# Optimizer

We have written our own optimizer objective function class and used the LBFGSpp module to minimize the error. At line 189 seen from the figure below we have limited the iteration to improve the performance. In addition, we have also tracked the lowest errors inside our **CubicSmileObjective** class inside **CubicSmile.cpp**. The main idea is to use the forward finite difference to obtain the gradient to find the minimum point, the cost function is still using the CalculateFittingError function.

A screen shot of a computer program

Description automatically generated

We have also constructed the smile with the best parameters we have found so far even if we meet any exceptions in the optimisation process as seen from the below figure.  
A screenshot of a computer program

Description automatically generated

# Smile Plot

A graph with blue dots and orange lines

Description automatically generatedA graph with blue dots and numbers

Description automatically generated

As seen from the plot when we are closer to the ATM vol the fit was improved with optimisation but after the range of the snapshot it seems that the error will increase which is understandable as there are less points to interpolate and the fact that we set the vol to the boundary values instead of extrapolating it.