# Certificate in Quantitative Finance

# Final Project Brief

## January 2024 Cohort

This document outlines topics available for this cohort. No other topics can be submitted. Each topic has by-step instructions to give you a structure (not limit) as to what and how to implement.

Marks earned will strongly depend on your coding of numerical techniques and presentation of how you explored and tested a quantitative model (report in PDF or HTML). Certain numerical methods are too involved or auxiliary to the model, for example, do not recode optimisation or RNs generation. Code adoption allowed if the code fully modified by yourself.

A capstone project requires own study and ability to work with documentation on packages that implement numerical methods in your coding environment e.g., Python, R, Matlab, C#, C++, Java. You do not need to pre-approve the coding language and use of libraries, including very specialised tools such as Scala, kdb+ and q. However, software like EViews is not coding.

Exclusively for current CQF delegates. No distribution.

To complete the project, you must code the model(s) and its numerical techniques form one topic from the below options and write an analytical report. If you continue from a previous cohort, please review topic description because tasks are regularly reviewed. It is not possible to submit past topics.

1. Credit Spread for a Basket Product (CR)
2. Deep Learning for Financial Time Series (DL)
3. Pairs Trading Strategy Design & Back test (TS)
4. Portfolio Construction using Black-Litterman Model and Factors (PC)
5. Optimal Hedging with Advanced Greeks (DH)
6. Blending Ensemble for Classification (ML)
7. Algorithmic Trading for Reversion and Trend-Following (AL)
8. Deep Neural Networks for Solving High Dimensional PDEs (DN)

Topics List for the current cohort will be available on the relevant page of Canvass Portal.

## Project Report and Submission

- First recommendation: do not submit Python Notebook ' as is' – there is work to be done to transform it into an analytical report. Remove printouts of large tables/output. Write up mathematical sections (with LaTeX markup). Write up analysis and comparison for results and stress-testing (or alike). Explain your plots. Think like a quant about the computational and statistical properties: convergence/accuracy/variance and bias. Make a table of the numerical techniques you coded/utilised.
- Project Report must contain sufficient mathematical model(s), numerical methods and an adequate conclusion discussing pros and cons, further development.
- There is no set number of pages. Some delegates prefer to present multiple plots on one page for comparability, others choose more narrative style.
- It is optimal to save Python Notebook reports as HTML but do include a PDF with page numbers – for markers to refer to.
- Code must be submitted and working.

FILE 1. For our download and processing scripts to work, it is necessary to name and upload the project report as ONE file (pdf or html) with the two-letter project code, followed by your name as registered on CQF Portal.
Examples: TS John Smith REPORT.pdf or PC Xiao Wang REPORT.pdf

FILE 2. All other files, code and a pdf declaration (if not the front page) must be uploaded as additional ONE zip file, for example TS John Smith CODE.zip. In that zip include converted PDF, Python, and other code files. Do not submit unzipped .py, .cpp files as cloud anti-virus likely to flash red on our side.
Do not submit files with generic names, such as CODE.zip, FinalProject.zip, Final Project Declaration.pdf, etc. Such files will be disregarded.

Submission date for the project is Thursday 22nd August 2024, 23.59 BST

There is no extension time to Final Project.

Projects without a hand-signed declaration or working code are incomplete.

Failure to submit ONE report file and ONE zip file according to the naming instructions means such a project will miss an allocation for grading.

All projects are checked for originality. We reserve an option of a viva voce before the qualification to be awarded.

## Project Support

### Advanced Electives

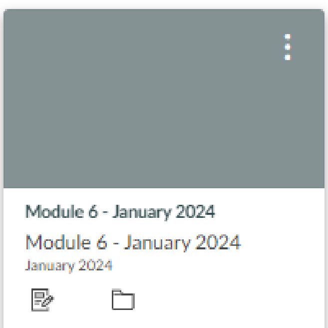
To gain background knowledge in a focused way, we ask you to review two Advanced Electives. Electives canvass knowledge areas and can be reviewed before/at the same time/closer to writing up Analysis and Discussion (explanation of your results).

- There is no immediate match between Project Topics and Electives
- Several workable combinations for each Project Topic are possible.
- One elective learning strategy is to select one `topical elective' and one `coding elective.'

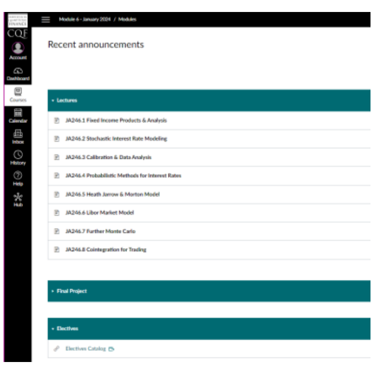
To access the electives:

Login to the CQF Learning Hub

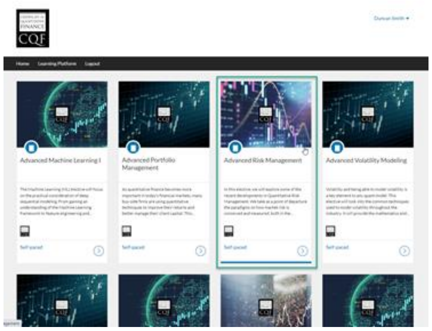
Navigate to Module 6 on your Dashboard.



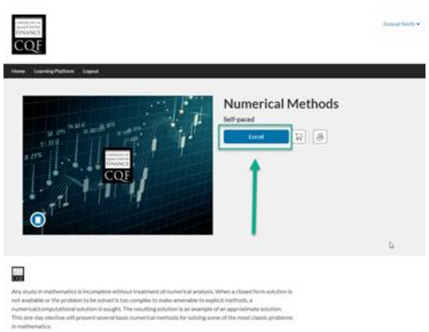

Click on *Electives* button on global navigation menu.



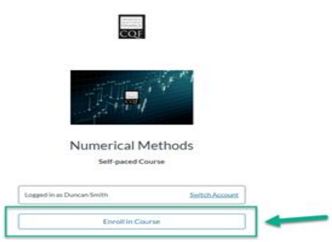

Scroll down to *Electives*, then click the *Electives Catalog*.



You will be redirected to the electives Catalogue, where you can view and review all electives available to you. Full descriptions for each elective can be found here.

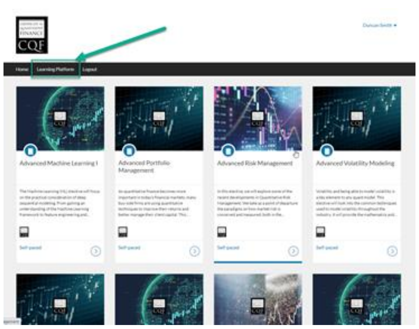



When on an elective click the *enrol* button




You will see the confirmation page, click the *enrol in Course* button to confirm your selection.

You will land on the successful enrolment page, where you can click to start the elective or return to the catalogue page.



When on the catalogue page you can click the *Learning Platform* link to return to Canvas. Your electives selected will appear on your learning dashboard.

## Workshop & Tutorials

Each project title is supported by a faculty member alongside a set of project workshops and tutorials.

| DATE | TITLE | TIME |
|---|---|---|
| 06/07/2024 | Final Project Workshop I (CR, PC & DH) | 13:00 – 15:30 BST |
| 13/07/2024 | Final Project Workshop II (TS, DL, ML, AL, DN & DH) | 13:00 – 15:30 BST |
| 19/07/2024 | Final Project Tutorial I (CR Topic) | 18:00 – 19:00 BST |
| 22/07/2024 | Final Project Tutorial II (TS, DH & AL Topic) | 18:00 – 19:00 BST |
| 23/07/2024 | Final Project Tutorial III (DL & ML Topic) | 18:00 – 19:00 BST |
| 24/07/2024 | Final Project Tutorial IV (PC & DN Topic) | 18:00 – 19:00 BST |

Faculty Support

Lead: Riaz Ahmad
Title: Credit Spread for a Basket Product (CR)
Project Code: CR

Lead: Kannan Singaravelu
Title: Deep Learning for Financial Time Series (DL)
Ensemble for Classification (ML)
Project Code: DL & ML

Faculty Lead: Richard Diamond
Title: Pairs Trading Strategy Design & Backtest (TS)
Optimal Hedging with Advanced Greeks (DH)
Algorithmic Trading for Reversion and Trend-Following (AL)
Project Code: TS, DH, AL

Faculty Lead: Panos Paras
Title: Portfolio Construction using Black-Litterman Model and Factors
Deep Neural Networks for Solving High Dimensional PDEs (DN)
Project Code: PC & DN

To ask faulty a question on your chosen topic, please submit a support ticket by clicking on the Support button which can be found in the bottom hand right corner on your portal.

## Coding for Quant Finance

- Choose programming environment that has appropriate strengths and facilities to implement the topic (pricing model). Common choice is Python, Java, C++, R, Matlab. Exercise judgement as a quant: which language has libraries to allow you to code faster, validate easier.

- Use of R/Matlab/Mathematica is encouraged. Often there a specific library in Matlab/R gives fast solution for specific models in robust covariance matrix/cointegration analysis tasks.

- Project Brief give links to nice demonstrations in Matlab, and Webex sessions demonstrate Python notebooks {does not mean your project to be based on that ready code.

- Python with pandas, matplotlib, sklearn, and tensorow forms a considerable challenge to Matlab, even for visualization. Matlab plots editor is clunky, and it is not that difficult to learn various plots in Python.

- ‘Scripted solution' means the ready functionality from toolboxes and libraries is called, but the amount of own coding of numerical methods is minimal or non-existent. This particularly applies to Matlab/R.

- Projects done using Excel spreadsheet functions only are not robust, notoriously slow and do not give understanding of the underlying numerical methods. CQF-supplied Excel spreadsheets are a starting point and help to validate results but coding of numerical techniques/use of industry code libraries is expected.

- The aim of the project is to enable you to code numerical methods and develop model prototypes in a production environment. <u>Spreadsheets-only or scripted solutions are below the expected standard for completion of the project.</u>

- What should I code? Delegates are expected to re-code numerical methods that are central to the model and exercise judgement in identifying them. Balanced use of libraries is at <u>own discretion</u> as a quant.

- Produce a small table in report that lists methods you implemented/adjusted. If using ready functions/borrowed code for a technique, indicate this and describe the limitations of numerical method implemented in that code/standard library.

- It is up to delegates to develop their own test cases, sensibility checks and validation. It is normal to observe irregularities when the model is implemented on real life data. If in doubt, reflect on the issue in the project report.

- The code must be thoroughly tested and well-documented: each function must be described, and comments must be used. Provide instructions on how to run the code.

# Credit Spread for a Basket Product

Price a fair spread for a portfolio of CDS for 5 reference names (Basket CDS), as an expectation over the joint distribution of default times. The distribution is unknown analytically and so, co-dependent uniform variables are sampled from a copula and then converted to default times using a marginal term structure of hazard rates (separately for each name). Copula is calibrated by estimating the appropriate default correlation (historical data of CDS differences is natural candidate but poses market noise issue). Initial results are histograms (uniformity checks) and scatter plots (co-dependence checks). Substantial result is sensitivity analysis by repricing.

A successful project will implement sampling from both, Gaussian and t copulae, and price all k-th to default instruments (1st to 5th). Spread convergence can require the low discrepancy sequences (e.g., Halton, Sobol) when sampling. Sensitivity analysis *wrt* inputs is required.

## Data Requirements

Two **separate** datasets required, together with matching discounting curve data for each.

1. **A snapshot of credit curves** on a particular day. A debt issuer likely to have a USD/EUR CDS curve – from which a term structure of hazard rates is bootstrapped and utilised to obtain exact default times, $u_i \to \tau_i$. In absence of data, spread values for each tenor can be assumed or stripped visually from the plots in financial media. The typical credit curve is concave (positive slope), monotonically increasing for $1Y, 2Y, \ldots, 5Y$ tenors.

2. **Historical credit spreads time series** taken at the most liquid tenor $5Y$ for each reference name. Therefore, for five names, one computes $5 \times 5$ default correlation matrix. Choosing corporate names, it is much easier to compute correlation matrix from equity returns.

   Corporate credit spreads are unlikely to be in open access; they can be obtained from Bloomberg or Reuters terminals (via your firm or a colleague). For sovereign credit spreads, time series of ready bootstrapped $\text{PD}_{5Y}$ were available from DB Research, however, the open access varies. Explore data sources such as `www.datagrapple.com` and `www.quandl.com`.

   Even if $\text{CDS}_{5Y}$ and $\text{PD}_{5Y}$ series are available with daily frequency, the co-movement of daily changes is market noise *more* than correlation of default events, which are rare to observe. Weekly/monthly changes give more appropriate input for default correlation, however that entails using 2-3 years of historical data given that we need at least 100 data points to estimate correlation with the degree of significance.

**If access to historical credit spreads poses a problem remember, default correlation matrix can be estimated from historic equity returns or debt yields.**

## Step-by-Step Instructions

1. For each reference name, bootstrap implied default probabilities from quoted CDS and convert them to a term structure of hazard rates, $\tau \sim Exp(\hat{\lambda}_{1Y}, \ldots, \hat{\lambda}_{5Y})$.

2. Estimate default correlation matrices (near and rank) and d.f. parameter (ie, calibrate copulæ). You will need to implement pricing by Gaussian and t copulæseparately.

3. Using sampling form copula algorithm, repeat the following routine (simulation):

   (a) Generate a vector of correlated uniform random variable.

   (b) For each reference name, use its term structure of hazard rates to calculate exact time of default (or use semi-annual accrual).

   (c) Calculate the discounted values of premium and default legs for every instrument from *1st* to *5th*-to-default. Conduct MC separately or use one big simulated dataset.

4. Average premium and default legs across simulations separately. Calculate the fair spread.

## Model Validation

- The fair spread for k*th*-to-default Basket CDS should be less than k-1 to default. Why?

- Project Report on this topic should have a section on **Risk and Sensitivity Analysis** of the fair spread *w.r.t.*

  1. default correlation among reference names: either stress-test by constant high/low correlation or ± percentage change in correlation from the actual estimated levels.
  2. credit quality of each individual name (change in credit spread, credit delta) as well as recovery rate.

  Make sure you discuss and compare sensitivities for all five instruments.

- Ensure that you explain historical sampling of default correlation matrix and copula fit (uniformity of pseudo-samples) – that is, Correlations Experiment and Distribution Fitting Experiment as will be described at the Project Workshop. Use histograms.

## Copula, CDF and Tails for Market Risk

The recent practical tutorial on using copula to generate correlated samples is available at: `https://www.mathworks.com/help/stats/copulas-generate-correlated-samples.html`
Semi-parametric CDF fitting gives us percentile values with fitting the middle and tails. Generalised Pareto Distribution applied to model the tails, while the CDF interior is Gaussian kernel-smoothed. The approach comes from Extreme Value Theory that suggests correction for an Empirical CDF (kernel fitted) because of the tail exceedances.

`http://uk.mathworks.com/help/econ/examples/using-extreme-value-theory-and-copulas-to-evaluate-market-risk.html`

`http://uk.mathworks.com/help/stats/examples/nonparametric-estimates-of-cumulative-distribution-functions-and-their-inverses.html`

# Deep Learning for Asset Prediction

**Summary**

Trend prediction has drawn a lot of research for many decades using both statistical and computing approaches including machine learning techniques. Trend prediction is valuable for investment management as accurate prediction could ensure asset managers outperform the market. Trend prediction remains a challenging task due to the semi-strong form of market efficiency, high noise-to-signal ratio, and the multitude of factors that affect asset prices including, but not limited to the stochastic nature of underlying instruments. However, sequential financial time series can be modeled effectively using sequence modeling approaches like a recurrent neural network.

**Objective**

Your objective is to produce a model to predict positive moves (up trend) using the Long Short-Term Memory Networks. Your proposed solution should be comprehensive with the detailed model architecture, evaluated with a backtest applied to a trading strategy.

- Choose one ticker of your interest from the index, equity, ETF, crypto token, or commodity.
- Predict trend only, for a short-term return (example: daily, 6 hours). Limit prediction to binomial classification: the dependent variable is best labelled [0, 1]. Avoid using [-1, 1] as class labels.
- Analysis should be comprehensive with detailed feature engineering, data pre-processing, model building, and evaluation.

**Note:** You are free to make study design choices to make the task achievable. You may redefine the task and predict the momentum sign (vs return sign) or direction of volatility. Limit your exploration to ONLY one asset. At each step, the process followed should be expanded and explained in detail. Merely presenting python codes without a proper explanation shall not be accepted. The report should present the study in a detailed manner with a proper conclusion. Code reproducibility is a must and the use of modular programming approaches is recommended. Under this topic, you do not recode existing indicators, libraries, or optimization to compute neural network weights and biases.

## Step-by-Step Instructions

1. The problem statement should be explicitly specified without any ambiguity including the selection of underlying assets, datasets, timeframe, and frequency of data used.

- If predicting short-term return signs (for the daily move), then training and testing over up to 5 years should be sufficient. If you attempt the prediction of 5D, 10D return for equity or 1W, 1M for the Fama French factor, you'll have to increase the data required to at least 10 years.

2. Perform exhaustive Feature Engineering (FE).

- FE should be detailed including the listing of derived features and specification of the target/label. Devise your approach on how to categorize extremely small near-zero returns (drop from the training sample or group with positive/negative returns). The threshold will strongly depend on your ticker. Example: small positive returns below 0.25% can be labelled as negative.
- Class imbalances should be addressed - either through model parameters or via label definition.
- Use of features from cointegrated pairs and across assets is permitted but should be tactical about design. There is no one recommended set of features for all assets; however, the initial feature set should be sufficiently large. Financial ratios, advanced technical indicators including volatility estimators, and volume information can be a predictor for price direction.
- OPTIONAL Use of news heatmap, credit spreads (CDS), historical data for financial ratios, history of dividends, purchases/disposals by key stakeholders (director dealings) or by large funds, or Fama-French factor data can enhance your prediction and can be sourced from your professional subscription.

3. Conduct a detailed Exploratory Data Analysis (EDA).

- EDA helps in dimensionality reduction via a better understanding of relationships between features and uncovers underlying structure, and invites detection/explanation of the outliers. The choice of feature scaling techniques should be determined by EDA.

4. Proper handling of data is a must. The use of a different set of features, lookback length, and datasets warrant cleaning and/or imputation.

5. Feature transformation should be applied based on EDA.

- Multi-collinearity analysis should be performed among predictors.
- Multi-scatter plots presenting relationships among features are always a good idea.
- Large feature sets (including repeated kinds, and different lookbacks) warrant a reduction in dimensionality in features. Self Organizing Maps (SOM), K-Means clustering, or other methods can be used for dimensionality reduction. Avoid using Principal Component Analysis (PCA) for non-linear datasets/predictors.

6. Perform extensive and exhaustive model building.

- Design the neural network architecture after extensive and exhaustive study.
- The best model should be presented only after performing the hyperparameter optimization and compared with the baseline model.
- The choice and number of hyperparameters to be optimized for the best model are design choices. Use experiment trackers like MLFlow or TensorBoard to present your study.

7. The performance of your proposed classifier should be evaluated using multiple metrics including backtesting of the predicted signal applied to a trading strategy.

- Investigate the prediction quality using AUC, confusion matrix, and classification report including balanced accuracy (if required).
- Predicted signals should be evaluated by applying them to a trading strategy.

* * *

# Pairs Trading Strategy Design & Backtest

Cointegrated relationship between prices opens way to build arbitrage around the mean-reversion in cointegrated residual. The special stationarity tests were developed in statistics, and we extend the standard cointegration testing with a controlling model for mean-reversion (OU process, aka Vasicek one-factor SDE in quant finance). Put signal generation and backtest at the centre of your effort: this project is not about the one-off run of statistical routines. Conventionally, pairs trading done via correlation; however, trading with 100%, -100% weights does not account for dollar difference and is not conductive to the stationarity of residual. Asset prices must be tied in using the error-correction model, which assumes prices as I(1) processes.

The numerical techniques to implement: matrix form autoregression, Engle-Granger procedure for error correction model, mean-reversion evaluation (theta, half-life), and optimal $Z^*$. You are encouraged to venture into the use of multivariate cointegration (Johansen procedure) and robustness of cointegration weights, ie, by adaptive estimation of your regression parameters (optionally). Strategy weights of multivariate coint will be difficult to attribute from the outside. The practical problems are certain loss $P\&L$ attribution for yourself, beta Python statsmodels on VECM. Engle-Granger procedure is very affine, therefore a good starting choice.

## Signal Generation and Backtesting

- Consider economic, quantiatative and even-driven reasons for the cointegration to persist. For example, a company and its potential aquisition target are particularly likely to form a robust coint relationship. Be inventive beyond equity pairs: consider commodity futures, VIX futures, within and across UST / UK Gilt yields.
- There are studies that filter the entire market for cointegration, but you can take more targeted approach, eg use the above, check for high correlation in search of pairs.
- Arb is realised by making entry/exit decisions with $\boldsymbol{\beta}_{Coint}$ as allocations $\boldsymbol{w}$. This is a natural long-short portfolio that generates a mean-reverting spread. All projects should include (a) trading signal generation from OU process fitting and (b) backtesting techniques, typically drowdown plots, rolling SR, and rolling beta.
- Does cumulative P&L behave as expected for a statistical arb trade? Is P&L coming from a few or many trades, what is the half-life? Maximum Drawdown and behaviour of volatility/VaR?

## Step-by-Step Instructions

### Part I: Pairs trade design. Preparatory cointegration analysis

1. Recode regression estimation in matrix form as an exercise. Can implement Vector Autogregression specification tests, which are (a) stability check with eigenvalues, and (b) identifying optimal lag $p$ with AIC BIC tests. However, these apply only for structural models between stationary changes; we are not forecasting returns in this project.
2. Implement Engle-Granger procedure and run it for each pair – later you can split the dataset into several periods and present more dynamic results. For Step 1, use the Aug-

mented DF with lag=1. For Step 2, write a short analysis, eg analyse the significance of EC-term, has it stayed the same each period.

3. We extend the procedure with 'Step 3' (not in the original Engle-Granger) mean-reversion evaluation. That allows trade design: enter on bounds $\mu_e \pm Z\sigma_{eq}$ and exit on $e_t$ reverting to the level $\mu_e$.

4. Do not assume $Z = 1$, implement some kind of optimisation, or simple iterative choice: vary $Z$ upwards and downwards in increments – and produce a chart or table analysing $N_{trades}$ for each level of $Z^*$. This is done to explicate the trade-off: wider bounds give the highest cumulative P&L with low $N_{trades}$, however those trades will be when $e_t$ deviates well-above or -below the level $\mu_e$. That itself might signal the likelihood of a structural break.

5. The formal investigation of structural breaks, particualrly within the reduced rank multivariate relationships, is beyond the scope of project. Testing for structural breaks is more an art than science. However, present a short discussion of your thought: reasons for and how your chosen asset coint relation might break apart.

If comfortable with R, can optionally utilise the ready multivariate analysis (package *urca*) to identify your cointegrated cases, where your assets have a term structure. Overall, your study should consider 2-3 different pairs.

**Part II: Backtesting**

Below items are recommendations, you can implement all or only a part of them as suitable to your asset classes and pairs. Overall, a 2-year period is considered to be minimum backtest, subject to the realities of cointegrated situation.

6. Think of ML/scikit-learn inspired backtesting: splitting train/test subsets and other cross-validaiton as feasible for financial time series (FTS).

7. Perform the systematic backtesting of your trading strategy. Take returns from a pairs trade (already with $Z^*$): produce drawdown plots, rolling Sharpe Ratio, at least one rolling beta with regard to S&P500 excess returns. Discuss your plots.

8. Cointegration analysis has the advantage of offering stable $\beta'_{Coint}$: delivering the stationary spread over 3-6 months without the need to be updated. That might not be a realistic assumption for FTS, for example, the industry sector might experience change in valuation multiples. Discuss benefits and disadvantages of regular re-estimation of cointegrated relationships: 5-8 months rolling window shifting by 10-15 days.

   OPTIONAL Would you consider Kalman filter/unscented particle filter for adaptive estimation? You can present the scheme of how coint residual regression equation changes under the adaptive estimation. One reference: `www.thealgoengineer.com/2014/online_linear_regression_kalman_filter/`.

**TS Project Workshop, Cointegration Lecture and Pairs Trading tutorial are your key resources.**

# Portfolio Construction using Black-Litterman Model and Factors

## Summary

Construct a factor-bearing portfolio, compute at least two kinds of optimisation. Within each optimisation, utilise the Black-Litterman model to update allocations with absolute and relative views. Compute optimal allocations for three common levels of risk aversion (Trustee/Market/Kelly Investor). Implement systematic backtesting: which includes both, regressing results of your portfolio on factors and study of the factors themselves (wrt the market excess returns).

Kinds of optimisation: mean-variance, Max Sharpe Ratio, higher-order moments (min coskewness, max cokurtosis) – implement <u>at least two</u>. Min Tracking Error also possible but for that your portfolio choice will be measured against a benchmark index. Computation by ready formula or specialised for quadratic programming. Adding constraints improves robustness: most investors have margin constraints / limited ability to borrow / no short positions.

OPTIONALLY, Risk Contributions can also computed *ex ante* for any optimal allocation, whereas computing ERC Portfolio requires solving a system of risk budget equations (non-linear). ERC computation is not an optimisation, however can be 'converted' into one – sequential quadratic programming (SQP).

### Portfolio Choice and Data

The choice of portfolio assets must reflect optimal diversification – according to your own understanding and approach. The simplest technical choice is choosing assets with the least correlation. For exposure/tilts to factor(s) – you need factor betas *a priori*, and include assets with either high or low beta, depending on the purpose specified by you.

A naive portfolio of S&P500 large caps can be said to be exposed to one factor, which is insufficient. A specialised portfolio for an industry, emerging market, credit assets should have 5-15 names (guide number), and $> 3$ notionally uncorrelated assets, such as commodity, VIX, bonds, credit spreads, real estate.

Factor time series (or several) can represent your uncorrelated asset – including factors as assets at the start and giving them preferential BL views is one way of systemic implementation of factor tilts.

- Mean-variance optimisation was specified by Harry Markowitz for simple returns (not log) which are *in excess* of the $r_f$. For risk-free rate, 3M US Treasury from pandas FRED dataset/ECB website rates for EUR/some small constant rate/zero rate – all are acceptable. Use 2-3 year sample, which means $> 500$ daily returns.
- Source for prices data is Yahoo!Finance (US equities and ETFs). Use code libraries to access that, Google Finance, Quandl, Bloomberg, Reuters and others. If benchmark index not available, equilibrium weights computed from the market cap (dollar value).
- In this variation of PC topic, it is necessary to introduce 2-3 factor time series and treat them as investable assets (5 Fama-French factors). If using Smart Beta ETFs present on their structure – you might find there is no actual long/short factors, just a long-only collection of assets with particularly high betas.

### Step-by-Step Instructions

**Part I: Factor Data and Study(Backtesting)**

1. Implement Portfolio Choice based on your approach to optimal diversification. Usually the main task is to select a few assets that gives risk-adjusted returns the same as/outperforms a much larger, naturally diversified benchmark such as S&P500. See Q&A document distributed at the Workshop.

2. Experiment which factors you are going to introduce, collect their time series data or compute.

   - The classic Fama-French factors are HML (value factor) and SMB (small business). RMW (robust vs. weak profitability) and CMA (conservative vs aggressive capex) are the new factors and you can experiment with them.
   - Exposure to sector or style can also be considered a factor.
   - It very recommended that you introduce an interesting, custom factor such as Momentum, BAB (betting against beta) – likely you will need to compute time series of its returns, however that can be as simple as returns from a short portfolio of top five tech stocks.

3. The range of portfolios, for which factors are backtested, is better explained at source `http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html`

4. Present P&L returns and Systematic Backtesting of your factors vs the Market (index of your choice), which includes performance, present plots of rolling beta and changing alpha. Ideally, you can present results for each factor beta independently and then, in combination. This work to be presented even before you engage in portfolio optimisation

**Part II: Comparative Analysis of BL outputs**

1. Plan your Black-Litterman application. Find a ready benchmark or construct the prior: equilibrium returns can come from a broad-enough market index. Implement computational version of BL formulae for the posterior returns.

2. Imposing too many views will make seeing impact of each individual view difficult.

3. Describe analytically and compute optimisation of **at least two kinds**. Optimisation is improved by using sensible constraints, eg, budget constraint, 'no short positions in bonds' but such inequality constraints $\forall w_i > 0$ trigger numerical computation of allocations. e.

4. You will end up with multiple sets of optimal allocations, even for a classic mean-variance optimisation (your one of two kinds). Please make your own selection on which results to focus your Analysis and Discussion – the most feasible and illustrative comparisons.

   - Optimal allocations (your) vs benchmark for active risk. Expected returns (naïve) vs implied equilibrium returns (alike to Table 6 in BL Guide by T. Idzorek.)
   - BL views are not affected by covariance matrix – therefore, to compute allocations shifted by views (through Black-Litterman model) with naive or robust covariance is your choice.

- Three levels of risk aversion – it is recommended that you explore at least for classical Min Var optimisation.

5. There is no rebalancing task for the project, particularly because posterior BL allocations expected to be durable.
6. Compare performance of your custom portfolio vs factors and market (rolling beta), independently and jointly. OPTIONALLY, compare performance of your portfolio to $1/N$ allocations / Diversification Ratio portfolio / Naive Risk Parity kind of portfolio and perform the systematic backtesting of that portfolio *wrt* to factors.

# Optimal Hedging with Advanced Delta Modelling v2024

## Summary

In this topic, you first consider the simple volatility arbitrage under condition of future realised volatility being above the implied, $V_a > V_i$. The workings can be found in Understanding Volatility lecture solutions. Implement in code the delta replication: long option, short stock. Use European call option Black-Scholes formulae, high/low volatility values of your choice. Provide visibility into how Gamma affects the P&L over $\Delta t$. GBM evolution should utilise its numerical techniques and advanced Monte-Carlo with variance reduction and/or ready low discrepancy sequences. Not necessary to consider a portfolio of options or several assets.

Improvement in delta-hedging can be achieved by adjusting the naive Black-Scholes Delta. We choose Minimum Variance Delta approach which corrects delta with the expected change in $\sigma_{imp}$ as a result of change in asset $S_t$,which is $\frac{\partial E(\sigma_{imp})}{\partial S}$. The underlying idea is simple: anticipate the magnitude of change in implied volatility and adjust Black-Scholes Delta for it.

MVD model coefficients $a, b, c$ estimation can be done via a regression on $\delta_{BS}, \delta_{BS}^2$ as independent variables directly, for which you re-arrange $\Delta V - \delta_{BS}\Delta S$ into dependent variable. More advanced method would be sequential least square programming optimiser (scipy) to solve for $a, b, c$ as parameters that minimise $\sum \epsilon_{MV}^2$, the hedging error defined in Hull and White (2017).

In your findings on MVD you are likely to make, illustrate and discuss the following discoveries. (1) You are likely to discover the MVD to be consistently less than BS Delta and therefore, naive delta-hedging typically overhedges. That is particularly effective for out-of-the-money call options. (2) After estimation, it can be uncovered that difference MV Delta - BS Delta ($\delta_{MV} - \delta_{BS}$) gives an inverted parabola-like function empirically (across delta buckets), and therefore quadratic fitting recipe arises. (3) The advanced implementation will consider the Gain function, defined as the percentage reduction in the sum of the squared residuals resulting from the hedge.

### IV Options Data

1. S&P500 index options data is available from brokerages and data providers **OptionsDX.com**, Polygon.io. OptionsDX is a particularly good current choice (2023) that provides some free End Of Day (EOD) option quotes. For each trading day, and each expiry/strike, you will need BS option price as implied vol percentage, delta, and vega: $(V, \delta_{BS}, \nu_{BS})_t$.

2. In the absence of historical options data, implied volatility values can be randomly generated from a **uniform** distribution from a range of values. To simulated values, apply a multiplicative factor that varies from short to long expiry. That recipe mimics IV patterns observed in equity index options, and can be adapted with your own knowledge of how smile transpires in markets you work with.

3. This project does not require fitted volatility surfaces or Bloomberg/front-desk level option price data but you can certainly use volatility data from any source.

### Step-by-Step Instructions

**Part I: Volatility Arb with improved GBM and Monte-Carlo**

1. Consider improvements to GBM asset evolution (Euler-Maruyana/Milstein schemes). Optionally, can consider modelling asset with jumps, eg, Merton jump diffusion, without going into stochastic volatility, eg Heston-Nandi. Variance Gamma is also relevant but suited for single-name assets with extreme movements.
   - consider MC variance reduction techniques, such as antithetic variates;
   - best practice is low discrepancy sequences, eg Sobol with the Brownian bridge.
2. Under the condition of known future realised volatility $V_a > V_i$, analytically and with Monte-Carlo confirm the items below. Report with both, complete mathematical workings to fold $P\&L_t$ and simulations of $P\&L_t$.
   - confirm *actual* volatility hedging leads to the known total P&L;
   - confirm and demonstrate *implied* volatility hedging leads to uncertain total, path-dependent P&L, and characterise on which parameters/Greeks it depends.
3. Think of additional analysis: consider how P&L decomposes in terms of Greeks. What is the impact of time-dependent Gamma $\Gamma_t$? What about $r^2 - \sigma_{imp}\delta t$? Consider findings from Part II MVD modelling, what are the implications of hedging with the smaller delta?

**Part II: Minimum Variance Delta**

1. begin with sorting your IV data – or each trading day, you will need BS option price as implied vol percentage, delta, and vega: $(V_t, \delta_{BS}, \nu_{BS})$. The term structure for option expiry $1M, 3M, 6M, 9M, 12M$, weekly expiries not necessary. Key choice to make here, if you are going to study Delta for out of the money call strikes, in addition to about ATM buckets $0.45 < \delta_{BS} < 0.55$ – each strike means a separate $a, b, c$ history for each expiry.
2. compute your dependent variable and run the fitting on $\delta_{BS}, \delta_{BS}^2$. Dependent side based on daily option price *changes* $\Delta V_t$, and you will need $(\Delta S_t, S_t)$ as well as Greeks noted above. The exact data columns will depend on how you organise regression or do SLSQP.
3. parameters $a, b, c$ can be constant for a study project, but rolling estimation itself is a calibration technique because for each expiry, you have time-dependent a,b,c (not 3 constants). Hull White recipe was 3M rolling window, then shift the start date by one day ($3 \times 22$ obs) – you can estimate with shorter/longer periods or shift by $5 - 10$ days. Also remember, option IVs can be simulated from a uniform distribution recipe.
4. For model validation, look at change of $a, b, c$ over time – we use regression as a fitting tool so they might not even be statistically significant. Check if $\delta_{MV} - \delta_{BS}$ gives an (inverted) parabolic shape, plot expected change in IV vs Delta.
5. $\mathbb{E}[\Delta\sigma_{imp}]$ expected to be negative but might not be. Is your achieved hedging Gain anywhere close to 15% and in which delta buckets and expiries?

**Additional Material – a small, curated collection of relevant articles will be distributed at Project Workshop (I or II). Volatility-related core lectures from Module 3 also support this Topic DH.**

# Blending Ensemble for Classification


**Summary**

Trend prediction is valuable for investment management as accurate prediction could ensure asset managers outperform the market. Trend predictions can be modeled effectively using machine learning algorithms; however, the choice of learning techniques to be adopted remains a challenging task. Ensemble learning is a powerful machine learning algorithm that combines the predictions of multiple machine learning models by mitigating the errors or biases that may exist in individual models by leveraging the collective intelligence of multiple models that leads to a more precise prediction.


**Objective**

Your objective is to produce a model to predict positive moves (up trend) using the Blending Ensemble technique. Your proposed solution should be comprehensive with the detailed model architecture, evaluated with a backtest applied to a trading strategy.

- Choose one ticker of your interest from the index, equity, ETF, crypto token, or commodity.
- Predict trend only, for a short-term return (example: daily, 6 hours). Limit prediction to binomial classification: the dependent variable is best labelled [0, 1]. Avoid using [-1, 1] as class labels.
- Analysis should be comprehensive with detailed feature engineering, data pre-processing, model building, and evaluation.
- Use only machine learning algorithms for this project.

**Note:** You are free to make study design choices to make the task achievable. You may redefine the task and predict the momentum sign (vs return sign) or direction of volatility. Limit your exploration to ONLY one asset. At each step, the process followed should be expanded and explained in detail. Merely presenting python codes without a proper explanation shall not be accepted. The report should present the study in a detailed manner with a proper conclusion. Code reproducibility is a must and the use of modular programming approaches is recommended. Under this topic, you do not recode existing indicators, libraries, or optimization algorithms.

## Step-by-Step Instructions

1. The problem statement should be explicitly specified without any ambiguity including the selection of underlying assets, datasets, timeframe, and frequency of data used.

- If predicting short-term return signs (for the daily move), then training and testing over up to 5 years should be sufficient. If you attempt the prediction of 5D, 10D return for equity or 1W, 1M for the Fama French factor, you'll have to increase the data required to at least 10 years.

2. Perform exhaustive Feature Engineering (FE).

- FE should be detailed including the listing of derived features and specification of the target/label. Devise your approach on how to categorize extremely small near-zero returns (drop from the training sample or group with positive/negative returns). The threshold will strongly depend on your ticker. Example: small positive returns below 0.25
- Class imbalances should be addressed - either through model parameters or via label definition.
- Use of features from cointegrated pairs and across assets is permitted but should be tactical about design. There is no one recommended set of features for all assets; however, the initial feature set should be sufficiently large. Financial ratios, advanced technical indicators including volatility estimators, and volume information can be a predictor for price direction.
- OPTIONAL Use of news heatmap, credit spreads (CDS), historical data for financial ratios, history of dividends, purchases/disposals by key stakeholders (director dealings) or by large funds, or Fama-French factor data can enhance your prediction and can be sourced from your professional subscription.

3. Conduct a detailed Exploratory Data Analysis (EDA).

- EDA helps in dimensionality reduction via a better understanding of relationships between features and uncovers underlying structure, and invites detection/explanation of the outliers. The choice of feature scaling techniques should be determined by EDA.

4. Proper handling of data is a must. The use of a different set of features, lookback length, and datasets warrant cleaning and/or imputation.

5. Feature transformation should be applied based on EDA.

- Multi-collinearity analysis should be performed among predictors.
- Multi-scatter plots presenting relationships among features are always a good idea.
- Large feature sets (including repeated kinds, and different lookbacks) warrant a reduction in dimensionality in features. Self Organizing Maps (SOM), K-Means clustering, or other methods can be used for dimensionality reduction. Avoid using Principal Component Analysis (PCA) for non-linear datasets/predictors.

6. Perform extensive and exhaustive model building.

- The architecture of a stacking model should involve at least three base learners.
- Hyperparameters of each base learners should be optimized.
- Type of base learners and meta model to be used are design choices.

7. The performance of your proposed classifier should be evaluated using multiple metrics including back-testing of the predicted signal applied to a trading strategy.

- Investigate the prediction quality using AUC, confusion matrix, and classification report including balanced accuracy.
- Predicted signals should be evaluated by applying them to a trading strategy.

* * *

# Algorithmic Trading for Reversion and Trend-Following

We consider algorithmic trading through the study of how to optimally run the basic strategies, and considering what affects their execution. The typical considerations are:

1. changes in price and volume (market impact);
2. timing and other order-specific events, subject to information about execution from API.

The real-time newsflow forms sentiment, and if you have access to sentiment indicator data, please incorporate that. However, it is more practically valuable to run linear regression, which estimates market impact, eg based on liquidity information.

Implement at least (a) one mean-version strategy with a specific indicator of your choice AND (b) one trend-following strategy with several indicators of your choice. You can code strategy backtest within Python on historical data (use higher frequency, eg 15 min). Brokers offer APIs as part of their offering to customers who wish to deploy their own trading algos. You can utilise use this project to code and test simple scripts for the Broker/API of your choice – Part II covers more detail. Real-time trading often means coding loops which check for buy/sell signal: even if API available in Python, better specialized and optimized language needed, such as GoLang. Your algorithmic trading code must handle exceptions, eg, various failures during the order execution and even inconsistent/incorrect information from the broker.

### Trend-Following Strategy

Trend-following strategies aim to capitalize on sustained movements in the market. The indicators such as the Moving Averages, Exponential Moving Averages (EMA) and Average Directional Index (ADX) to confirm trends. The ADX measures the strength of a trend, helping to filter out weak trends that may not be profitable. You can combine moving averages with ADX, in order to generate better quality trading signals.

Adjustment of algo trading parameters, such as resampling interval and averaging period has to be performed to assess the effectiveness of your specific trend-following approach. That adjustment has to be tested under various market conditions as well.

### Part I: Generic Strategies Made Proprietary

Write Python code that implements your core strategies – this can be done purely in Python on historical data, or you can proceed right away using Python + Broker API (see below).

1. For a trend-following strategy, common choices are Exponential Moving Averages (EMA) and Average Directional Index (ADX) type of oscillator. Also, convergence/divergence indicator, typically Moving Average Convergence/Divergence (MACD).
2. Another simple but practical trend-following indicator, that is primarily used in FX, is as follows:
   - Step 1: resample the prices at regular intervals (eg, 30 seconds); can use *pandas* resample method.

- Step 2: calculate an average price over the longer period (eg, past five-minute intervals). Implement trading logic to open a position when market prices deviate from that average, and close the position when prices revert back to it.
- Step 3: compute the ratio of the short-term average to the long-term average price. No trend is signaled by the ratio of 1, short-term prices $\approx$ the long-term prices. Uptrend signaled by the ratio above 1, and downtrend by less than 1.

3. Present full mathematical description of the trend-following indicators chosen. Adjust the ratios for <u>your traded assets and time periods</u> and discuss their optimality.
4. For a mean-version strategy, the formal modeling of mean-reversion with OU process can be invoked expecting the price to mean-revert. For that to generate P&L, it is very likely that unscented filtering/Kalman filtering (see Topic TS), however there is no requirement to implement filtering.
   - Common signal is simply $Z^*$ deviation from the price, or you can use a measure of distance implemented in machine learning.

Please note that for each strategy you need to decide on your own indicator, how it is computed and what constitutes a trading signal (eg, crossover of 20D Exponential Moving Average).

**Part II: Broker API**

1. The common API choice is REST (Representational State Transfer).
   - Alpaca, Interactive Brokers Web, and Oanda all have their own versions. In particular, Alpaca REST API is free and includes asynchronous events handling based on WebSocket and Server Side Events (SSE).
   - REST API is referred to as HTTP API because it utilizes HTTP methods, eg GET (retrieve data), POST (create new data), PUT (update data). Useable for non-time-critical operations such as retrieving historical data, account information, placing orders, and getting order status.
2. The more industrial strength and lower latency API choice is FIX (Financial Information eXchange). A messaging protocol designed for the **real-time** exchange of securities transactions. It supports submission and cancellation of various order types, trade execution reports, and market data dissemination – all for high frequency trading. FIX is used by large institutions, funds, and broker/dealers.
3. Interactive Brokers offer TWS API with connection to their client application and possibility to use C++, C#, Java, Python. Useful comparison of these API but not an endorsement is available at `www.interactivebrokers.com/en/index.php?f=5041`.
4. Treat this project as more professional in regard to the trading, eg, instead of Yahoo!Finance, fetch your data from OpenBB/brokerage and ensure it is of higher quality.

Describe order types suitable to your traded assets, and attempt to write code for handling those orders using the API. Set price parameters far from the market to avoid execution. It is an absolute recommendation though that you use a paper trading account.

**Part III: Evaluate Risk and Test Thrice**

Opened market position is subject to various types of risk. While the market risk can be managed using simple quant techniques such as rolling VaR computation, the order handling risk is more important and you would have attempted to address it in Part II.

1. Handling Events. The code must have routines on order handling, including re-verification of server responses. The order can be cancelled, partially filled or even returned with an incorrect information about the fill.
2. Positions Tracking. The code can rely on account updates in order to reconfirm a trade has been entered to/exited.
3. Market Data. There could be issues with market data input quality. Ideally the code must have simple validation checks.
4. Systematic Backtesting (eg, concentration in asset, Beta-to-SPY) are of no utility for this type of algo trading project, however for overall performance evaluation it's useful to provide a tearsheet with information about turnover. Compute Drawdowns, Sharpe Ratio and some form of Value-at-Risk, adapted for the short-term trading.

The general principle is to preserve the trading capital as much as possible.

# CQF Project: Deep Neural Networks for Solving High Dimensional PDEs in Quantitative Finance (DeepPDE)

## OVERVIEW

There is little doubt that deep learning techniques have revolutionized certain areas of computer science such as computer vision and natural language processing. However, the impact of deep learning and other machine learning techniques has been limited in quantitative finance, especially in traditional areas of quantitative finance such as option pricing, hedging and portfolio management. These traditional areas of quantitative finance are dominated by rigorous mathematical models, and pricing frameworks such as risk-neutral evaluation. In contrast computer science applications rely on high quality empirical data. As a result we have seen deep-neural networks be used to mainly solve regression problems in order to develop fast pricing engines (e.g. a neural network is given the prices of various contracts and learns the pricing function). However in the last few years we have observed a number of approaches that go beyond the simple function fitting/regression framework and several recent works have embraced the new set of tools. The objective of this project is to investigate the use of Deep-Neural Networks for solving high-dimensional PDEs that appear in pricing and risk management applications.

## METHODOLOGY

In this project you will develop a Deep Neural Network (DNN) that can solve the following class of $d$-dimensional Partial Differential Equations:

$$
\begin{aligned}
&\frac{\partial u}{\partial t} + \frac{1}{2}\mathrm{Tr}(\sigma(t,x)\sigma(t,x)^\top \nabla^2 u(t,x)) + \nabla u(t,x)^\top \mu(t,x) + f(t,x,u(t,x),\sigma(t,x)^\top \nabla u(t,x)) = 0 \\
&u(T,x_1,\ldots,x_d) = g(x_1,\ldots,x_d)
\end{aligned}
\tag{1}
$$

Where:

- $u(t,x)$ with $t \in \mathbb{R}_+$, and $x \in X \subset \mathbb{R}^d$ is the unknown function we need to compute, $\nabla u(t,x)$ is the gradient of the solution with respect to $x$ and $\nabla u(t,x)$ is the Hessian (again with respect to $x$).
- $\sigma(t,x)$ is a known $d \times d$ matrix valued function.
- Tr denotes the trace of a matrix and $T$ denotes a transpose.
- $f$ is a known nonlinear function
- $g$ is a known boundary condition.

While it is entirely possible to develop an approach to solve the general case in (1) your starting point for this project is the Black-Scholes PDE with $d$ uncorrelated assets and with a constant volatility and risk free rate. You are encouraged to first understand the provided code for this problem before extending it to another pricing problem. Consider the following special case of (1) that can be used to price European-style contracts,

$$
\begin{aligned}
&\frac{\partial u}{\partial t} + \sum_{i=1}^{d}\left(\frac{1}{2}\sigma_i^2 \frac{\partial^2 u}{\partial x_i^2} x_i^2 + r\frac{\partial u}{\partial x_i} x_i\right) - ru(t,x) = 0 \\
&u(T,x_1,\ldots,x_d) = g(x_1,\ldots,x_d)
\end{aligned}
\tag{2}
$$

Extensions to other type of contracts and other applications of the framework (e.g. for solving stochastic optimal control problems are given at the end of this document).

In this project we will focus on one of the most promising approaches to solving (2) that relies on a solving a forward-backward system of stochastic differential equations. The derivation of the equations below will also be explained in the project workshop.

$$\begin{aligned} \mathrm{d}X_t^i &= rX_t^i \mathrm{d}t + \sigma^i X_t^i \mathrm{d}W^i, \quad X_0^i = x^i, \ i = 1, \dots, d \\ \mathrm{d}Y_t &= rY_t \mathrm{d}t + \sum_{i=1}^{d} \sigma^i X_t^i Z_t^i \mathrm{d}W^i, \quad Y_T = g(X_T^1, \dots, X_T^d) \end{aligned} \tag{3}$$

The interpretation of (3) is as follows: $X^i$ denotes the price of asset $i$ and for simplicity we use a constant $r > 0$ risk free rate. Note that for simplicity we have also assumed that $\sigma^{i,j} = 0$ for $i \neq j$ and that the Brownian motions are all uncorrelated i.e. $\mathrm{cov}(\mathrm{d}W^i \mathrm{d}W^j) = 0$ for $i \neq j$. The second equation for $Y$ is a Backward Stochastic Differential Equation. The unknowns for the BSDE are $Y_t$ and $Z_t$. In terms of interpretation it can be shown that $Y_t = u_t$ and $Z_t = \nabla_x u(t, x)$. In other words $Y_t$ is the price of the option and $Z_t$ is the option's delta.

Since $Z_t$ is not known it is not possible to (easily) simulate the system in (2). Instead the idea is to parameterize the solution $u(t, x) \approx u(t, x; \Theta)$ where $\Theta$ are the parameters of a neural network, and compute $Z_t \approx Z_t(\Theta) = \nabla_x u(t, x; \Theta)$. The derivative $\nabla_x u(t, x; \Theta)$ is with respect to $x$ and can be computed with backpropagation.

### STEP-BY-STEP INSTRUCTIONS

Your code should have the following main steps.

1. Define a Neural Network to represent $u(t, x; \Theta)$. See the example code provided for an example of how to set this up using Pytorch.
2. Generate $M$ Monte Carlo paths using a simple Euler scheme to discretize the dynamics in (3). Since your Euler discretization will use $Z_t(\Theta)$ instead of the correct $Z_t$ you will need to optimize the parameters $\Theta$
3. Use an optimizer to optimize the Neural Network parameters. Define an appropriate loss function such as the one described in [2].

A minimum working example using the Pytorch library is provided along with the specification. The minimum working example prices vanilla call options, you should modify/extend the code to solve other problems. A successful project should: be able to demonstrate an extension of the provided code to high dimensions (e.g. $d = 100$) and (optionally) apply the framework to a different product class then the one provided. Below are some ideas and references you can use to extend your code.

- The working example is based on the work from [2]. In the paper you can find some ideas on how to train the model faster and some more stable architectures that work for very high dimensions $d = 100$.
- The article in [3] is also a good introduction to the topic and has a discussion on exotic options (still in one-dimension).
- The article in [4] reviews several architectures and algorithms for solving high-dimensional pricing problems.
- The article in [1] investigates the use of deep neural networks for XVAs.
- Feel free to explore the papers above and other references and consider extending the minimum working example to applications that are of intrest to you, e.g. introduce correlation between the assets, consider dynamics that are not log-normal, price exotics or XVAs, etc.

## References

[1] A. Gnoatto, A. Picarelli, and C. Reisinger. Deep xva solver: A neural network–based counterparty credit risk management framework. *SIAM Journal on Financial Mathematics*, 14(1):314–352, 2023.

[2] B. Güler, A. Laignelet, and P. Parpas. Towards robust and stable deep learning algorithms for forward backward stochastic differential equations. *33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.*, 2019.

[3] B. Hientzsch. Deep learning to solve forward-backward stochastic differential equations. *Risk Magazine, February*, 2021.

[4] J. Liang, Z. Xu, and P. Li. Deep learning-based least squares forward-backward stochastic differential equation solver for high-dimensional derivative pricing. *Quantitative Finance*, 21(8):1309–1323, 2021.

**ELECTIVES RECOMMENDATION -- CQF January 2024**

| | |
|---|---|
| Credit Spread for a Basket Product | Counterparty Credit Risk Modeling<br>Numerical Methods<br>R for Data Science and ML |
| Optimal Hedging with Advanced Greeks | Advanced Volatility Modeling<br>Numerical Methods |
| Portfolio Construction using Black-Litterman | Risk Budgeting for Asset Allocation<br>Advanced Portfolio Management<br>Behavioral Finance for Quants |
| Pairs Trading Strategy Design & Backtest | Algorithmic Trading Il<br>Numerical Methods |
| Deep Learning for Financial Time Series | Advanced Machine Learning I<br>Advanced Machine Learning II |
| Blending Ensemble for Classification | Advanced Ensemble Modeling<br>Advanced Machine Learning I<br>Advanced Machine Learning II |
| Algorithmic Trading for Reversion and Trend-Following | Algorithmic Trading I<br>Algorithmic Trading Il<br>Decentralized Finance Technologies |
| Deep Neural Networks for Solving High Dimensional | Advanced Machine Learning I<br>Advanced Machine Learning II<br>Advanced Volatility Modeling |

# Reading List for Final Project (Selected Topics)

The puprpose of this list is not to provide textbooks, but rather more specific chapters and sample articles, from which you will gain topic knowledge/key model knowledge. The readings identified below and additional curated titles will be released with Project Workshops and/or Project Tutorials in files named *Topic XX - Additional Material.zip*.

### Reading List: Credit Portfolio

- Very likely you will revisit *CDO & Copula Lecture* material, particularly slides 48-52 that illustrate Elliptical copula densities and discuss Cholesky factorisation.
- *Sampling from copula* algorithm is in *relevant Workshop* and *Monte Carlo Methods in Finance* textbook by Peter Jaekel (2002) – see Chapter 5.
- Rank correlation coefficients are introduced *Correlation Sensitivity Lecture* and P. Jaekel (2002) as well. CR Topic Q&A document gives the clarified formulae and explanations.

### Reading List: Portfolio Construction

- CQF Lecture on *Fundamentals of Optimization and Application to Portfolio Selection*
- *A Step-by-step Guide to The Black-Litterman Model* by Thomas Idzorek, 2002 tells the basics of what you need to implement.
- *The Black-Litterman Approach: Original Model and Extensions* Attilio Meucci, 2010. `http://ssrn.com/abstract=1117574`
- Marcenko-Pastur denoising / LW nonlinear shrinkage became optional. Either method aims to make the covariance matrix such that mean-variance optimisaiton is less perturbed by the degree of noise. Resources can be provided with the relevant FP Workshop/Tutorial.

### Reading List: Cointegrated Pairs

- *Modeling Financial Time Series*, E. Zivot & J. Wang, 2002 – we distribute Chapter 12 on Cointegration with the relevant Project Workshop (concerns with terms inside error-correction equation).
- Instead of a long econometrics textbook, read up *Explaining Cointegration Analysis: Parts I and II* by David Hendry and Katarina Juselius, 2000 and 2001. *Energy Journal*.
- Appendices of this work explain key econometric and OU process maths links, *Learning and Trusting Cointegration in Statistical Arbitrage* by Richard Diamond, *WILMOTT* `papers.ssrn.com/sol3/papers.cfm?abstract_id=2220092`.

## Reading List: Delta Hedging and Local Volatility

- *Optimal Delta Hedging for Options* by John Hull & Alan White. *Journal of Banking and Finance*, Vol 82, Sept 2017. Available on SSRN `papers.ssrn.com/sol3/papers.cfm?abstract_id=2658343`.
- *Manufacturing and Managing Customer-Driven Derivatives* textbook by Dong Qu – we distribute Chapter 12 (pp. 319-331) for the Local Volatility in Rates model (if the relevant Topic LV is offered).
- *Lecture 7: Local Volatility Continued* by Emmanuel Derman (2008) lectures on Volatility Smile, which we still find useful. Local Volatility in general can be read about in any good textbook.
- *Advanced Volatility Modelling* comes either as Elective or Alumni Workshop in CQF Lifelong Library (but it might concern with jump-diffusion and stochastic volatility).

## Reading List: Deep Learning and Ensemble Learning

- *Short-term stock market price trend prediction using a comprehensive deep learning system* by Jingyi Shen and M. Omair Shafiq, Journal of Big Data, Vol 7 (2020).
- *A graph-based CNN-LSTM stock price prediction algorithm with leading indicators* by Jimmy Ming-Tai Wu et al. (2021).
- *A comprehensive evaluation of ensemble learning for stock-market prediction* by Isaac Kofi Nti et al., Journal of Big Data, Vol 7 (2020).
- *A Blending Ensemble Learning Model for Crude Oil Price Prediction* by Mahmudul Hasan et al. (2022). `papers.ssrn.com/sol3/papers.cfm?abstract_id=4153206`.
- *Advanced Machine Learning I & II* comes either as Elective or Alumni Workshop in CQF Lifelong Library.