Individual Case Study

Digital Finance: Models, Markets, and Implementation

[Instructor Name]University Name¿University Name

Semester: [Term Year] • Due: 3 weeks after seminar

1 Overview

This case study is an **individual assignment** and the central assessment component of the MSc Digital Finance seminar. You will select one of the four case studies below and produce a self-contained research report together with a reproducible code repository.

- **Team size:** Individual (no group work).
- **Deliverables:**
  1. An 8-page written report (excluding references and appendix).
  2. A code repository (GitHub link or `.zip` archive).
- **Due date:** 3 weeks after seminar completion. Submit the report as PDF and the code repository via the course platform.
- **Weight:** 40% of the final grade.

2 Case Study Options

Choose **one** of the following four case studies.

2.1 Case Study 1: GARCH + Jump-Diffusion Pricing Model for ETH Options

Build a quantitative pricing model for ETH options that combines volatility clustering with jump risk.

(a) **GARCH estimation.** Fit a GARCH(1,1) model to ETH daily log-returns using maximum likelihood estimation via the `arch` Python package. Report parameter estimates (ω, α, β) with standard errors and verify $\alpha + \beta < 1$.

(b) **Jump-diffusion simulation.** Implement a Merton jump-diffusion model using Monte Carlo simulation with at least 10,000 sample paths. Calibrate jump intensity λ, mean jump size μ_J, and jump volatility σ_J from the data.

(c) **Option pricing.** Price a 1-month at-the-money (ATM) call option on ETH using both the GARCH-based approach and the jump-diffusion model. Report prices, implied volatilities, and Greeks (at minimum Δ and Γ).

(d) **Model comparison.** Compare model prices to market quotes from Deribit (or another options exchange). Use AIC/BIC or RMSE to evaluate model fit. Discuss which model better captures the implied volatility smile.

Report contents: model description and derivation, parameter estimates with standard errors, pricing results, model comparison metrics, and a discussion of limitations (e.g., discrete vs. continuous hedging, liquidity of crypto options).

2.2 Case Study 2: Uniswap V3 LP Profitability Analysis

Analyze the profitability of liquidity provision on Uniswap V3 using real on-chain data.

(a) **Pool selection.** Select 3 Uniswap V3 pools with different characteristics, e.g., ETH/USDC 0.3%, ETH/USDC 0.05%, and WBTC/ETH 0.3%.

(b) **Profitability decomposition.** For each pool, calculate impermanent loss, fees earned, and net returns for concentrated positions at various range widths ($\pm 5\%$, $\pm 10\%$, $\pm 20\%$, full range).

(c) **V3 vs. V2 comparison.** Compare V3 concentrated-liquidity positions to hypothetical V2 (full-range) positions in terms of capital efficiency, fee capture, and impermanent loss.

(d) **Breakeven analysis.** Determine the breakeven realized volatility for each range width: at what volatility level does fee income exactly offset impermanent loss?

Report contents: data sources and collection methodology, profitability results with charts, optimal range-width recommendations, and a risk assessment discussing rebalancing costs and tail scenarios.

2.3 Case Study 3: Crypto Portfolio with CVaR Constraints

Build and stress-test a cryptocurrency portfolio using modern risk measures.

(a) **Data collection.** Collect at least 2 years of daily returns for BTC, ETH, SOL, ADA, and a stablecoin (e.g., USDC, used as the risk-free proxy).

(b) **Distributional analysis.** Fit a Student-t distribution to each asset's returns. Estimate degrees of freedom ν and compare tail behaviour to the Gaussian assumption using Q-Q plots and a Jarque–Bera test.

(c) **Mean-variance frontier.** Construct the mean-variance efficient frontier. Report the tangency portfolio weights and Sharpe ratio.

(d) **CVaR optimization.** Build a CVaR-optimized portfolio at the 95% confidence level using linear programming (e.g., via `cvxpy` or `scipy.optimize`). Compare portfolio weights, expected return, and risk to the mean-variance solution.

(e) **Stress testing.** Simulate two historical stress scenarios—the March 2020 COVID crash and the November 2022 FTX collapse—and report portfolio drawdowns for both the MV and CVaR portfolios.

Report contents: methodology, distributional analysis with test statistics, portfolio weights and efficient frontiers, stress-test results, and a comparison of MV vs. CVaR approaches including when each is preferable.

2.4 Case Study 4: MEV Detection on Ethereum

Build an MEV (maximal extractable value) detection tool for the Ethereum blockchain.

(a) **Data collection.** Collect Ethereum transaction data for at least 1,000 consecutive blocks using the Etherscan API, a public dataset (e.g., Google BigQuery `bigquery-public-data.crypto_ethereum`), or an archive node.

(b) **Sandwich attack detection.** Identify sandwich attacks by pattern matching: a front-run transaction and a back-run transaction from the same address bracketing a victim swap in the same block and targeting the same pool.

(c) **MEV quantification.** For each detected sandwich, compute the MEV extracted (profit of the attacker) and aggregate per block. Report summary statistics: mean, median, and total MEV over the sample.

(d) **Statistical analysis.** Determine which DEX pools are most targeted, what victim trade sizes attract sandwiches, and whether MEV activity varies by time of day or gas price level.

Report contents: detection methodology and algorithm pseudocode, results with descriptive statistics and visualizations, false-positive discussion, and policy recommendations (e.g., private mempools, MEV redistribution mechanisms).

3 Grading Rubric

The case study is graded out of 100 points.

Criterion	Points	Description
Methodology	30	Is the approach sound? Are formulas correct? Are statistical methods (estimation, testing, optimization) properly applied?
Analysis quality	25	Are results thorough? Are charts clear and informative? Are edge cases and robustness checks addressed?
Interpretation	25	Do you explain what the results mean in economic or financial terms? Are limitations and caveats acknowledged honestly?
Presentation	20	Is the report well-structured and clearly written? Is the code clean, documented, and reproducible?

4 Report Guidelines

- **Length:** 8 pages maximum, excluding references and any appendix. Submissions exceeding 8 pages will be penalized.
- **Formatting:** 12pt font, 1.5 line spacing, margins of at least 2.5 cm.
- **Code submission:** All code must be submitted alongside the report—either as a GitHub repository link or a `.zip` archive containing Jupyter notebooks or `.py` files.
- **Reproducibility:** Include a `README` file explaining how to install dependencies and reproduce every figure and table in the report. Pin package versions (e.g., `requirements.txt`).
- **References:** Minimum 8 references. Use the course bibliography (`references-msc.bib`) as a starting point; additional sources are encouraged.
- **Citation style:** Author-year format (e.g., "Lehar and Parlour, 2025" or "Bollerslev, 1986"). Use a consistent style throughout.
- **Appendix (optional):** Extended derivations, additional robustness tables, or supplementary code may be placed in an appendix. The appendix does not count toward the page limit but will not be the primary basis for grading.

Academic integrity: All work must be your own. Cite all sources. The university's policy on plagiarism applies in full. Use of AI-assisted writing tools must be disclosed in a footnote on the first page; code generated with AI assistance must be clearly marked in comments.