4OPT1: Numerical Project in Python
Portfolio Selection

Elouan KERGOURLAY, Ramzi JEBALI

December 3, 2025

1 Introduction

The objective of this project is to implement and analyze numerical methods for the portfolio selection problem. We consider a market with n assets held over a fixed period. The decision variable is the portfolio vector $x \in \mathbb{R}^n$, where x_i represents the amount invested in asset i.

We assume a stochastic model for price changes, characterized by a mean vector μ and a covariance matrix Σ. The return of the portfolio is a random variable with expected value $\mu^\top x$ and variance $x^\top \Sigma x$. The goal of the optimization is to find a trade-off between maximizing the expected return and minimizing the risk (variance).

1.1 Markowitz portfolio problem

The classical formulation, introduced by Markowitz, is a convex Quadratic Programming (QP) problem. It seeks to minimize the portfolio variance subject to a minimum expected return $r_{\min}$, a total budget constraint (normalized to 1), and a no-short-selling constraint ($x \geq 0$).

In this report, we will solve this problem using convex optimization tools (CVXPY) and explore more complex variations involving probabilistic constraints and higher-order moments (Kurtosis), which require non-convex optimization techniques such as Sequential Quadratic Programming (SQP).

2 CVXPY and Basic Optimization

2.1 A simple least-squares problem

Question 1 First of all, we solve the initial constrained least-squares problem presented below.

$$\min_{0 \leq x \leq 1} \frac{1}{2} \|Ax - b\|_2^2. \tag{1}$$

The optimal value found for n=20 is: 4.453231.

Question 2 We analyzed the computational time of CVXPY as problem dimensions grow.

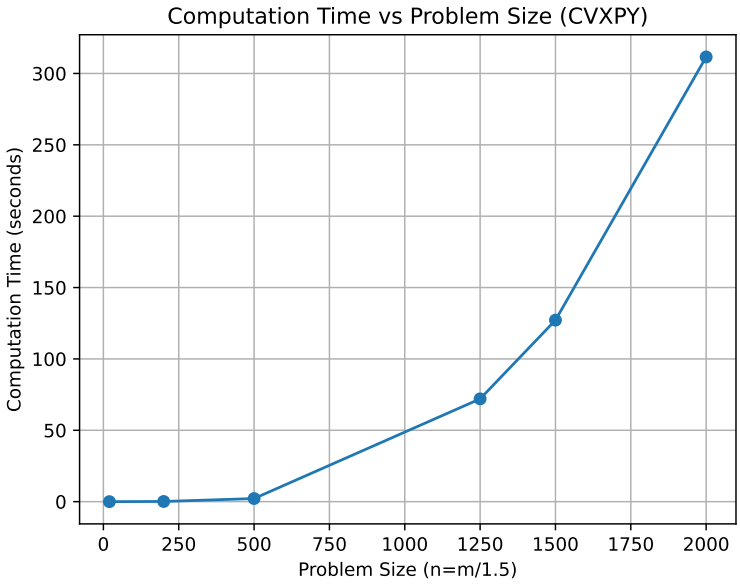

Figure 1: Computational time of CVXPY as a function of problem size.

Question 3 We compared computational times of CVXPY and SCIPY to solve the problem as a function of problem dimensions as they grow.

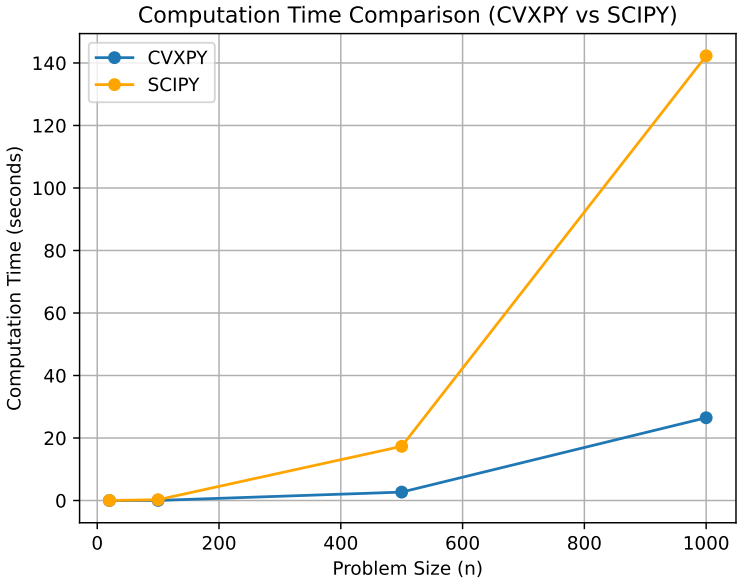

Figure 2: Computational time of CVXPY as a function of problem size.

The comparison between CVXPY and SCIPY (SLSQP) reveals that both solvers converge to the same global minimum.

We observe that `scipy.optimize.minimize` with the SLSQP solver is a lot slower than CVXPY as dimensions grow. Its computational time grow exponentially.

2.2 Solve the Markowitz portfolio problem

Question 4 We implemented the Markowitz portfolio problem and solved it using CVXPY.

```python
def markovitz_portfolio(mu, sigma):
    n = mu.shape[1] # dimension of the problem
    rmin = 1.6/252  # given value of the minimum return constraint

    x = cp.Variable((n,1))
    # (n,1) to force it to be a column vector as wanted by compute_metrics
    objective = cp.Minimize(cp.quad_form(x, sigma))
    constraints = [
        cp.sum(x) == 1,         # fully invested, use of total budget
        mu @ x >= rmin,         # minimum return constraint
        x >= 0                  # no short positions
    ]

    prob = cp.Problem(objective, constraints)
    prob.solve()

    return x.value
```

Listing 1: Markowitz Portfolio Solver

Question 5 We first evaluated the Markowitz portfolio on a small subset of 3 assets.

- **Training 2016-2019:** expected return of approximately 19.72% and a variance of 1.08. The model fits the historical data perfectly to meet the $r_{\min}$ constraint while minimizing risk.
- **Testing 2020:** The performance degrades significantly. The realized return drops to 0.36% and the variance increases to 5.49. This discrepancy highlights the stationarity gap: parameters estimated on pre-COVID data (2016-2019) fail to accurately capture the market regime of 2020, leading to poor out-of-sample performance.

Increasing the pool to $n = 20$ assets allows the solver to exploit lower correlations between a larger number of stocks.

We observe that the portfolio variance on the training set decreases from 1.08 to 0.43. This confirms the fact that risk is reduced by combining more assets.

However, we weren't able to mitigate the risk due to the market crash. It is still unstable out-of-sample, especially during crisis periods like 2020, regardless of the number of assets.

2.3 Probabilistic variant

Question 6 We implemented problem (3) that introduces a risk level $\beta = 0.49$ and a loss threshold α.

With the given values of the parameters, we got the same results, which means that the constraint was not active, ie the risk requirement set by the parameters was naturally fulfilled.

Question 7 By extending the testing window to include 2021, we observed significant improvements in the portfolio performance metrics:

The annualized return increased from 0.36% to 4.05%. The portfolio, which suffered heavily during the 2020 crash, benefited from the strong bullish trend of 2021, validating the importance of a longer investment horizon to smooth out short-term crises.

The portfolio's standard deviation dropped significantly from 37.21 to 28.31. The extreme volatility spikes of March 2020 are averaged out by the more stable regime of 2021. This confirms that evaluating risk on a single crisis year tends to overestimate the long-term risk profile of a strategy.

2.4 Plotting

Question 8 The time-series evolution of the selected assets is shown in Figure 3.

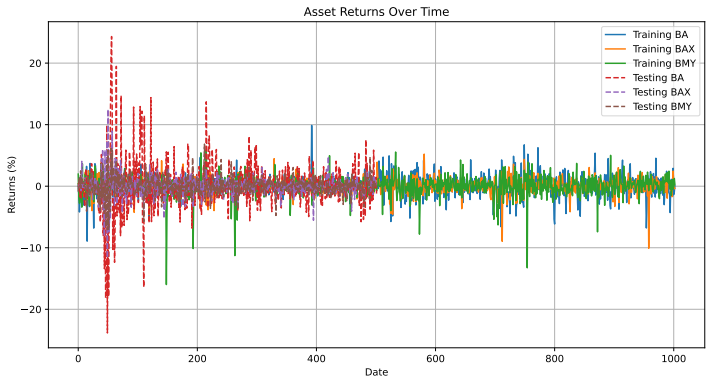

Figure 3: Time series of 3 assets (Train and Test).

Question 9 We performed a Monte Carlo analysis ($N = 1000$ samples). The scatter plot below shows the trade-off between standard deviation and return.

Figure 4: Monte Carlo simulation: Deviation vs Return (Comparison 3 vs 20 assets).

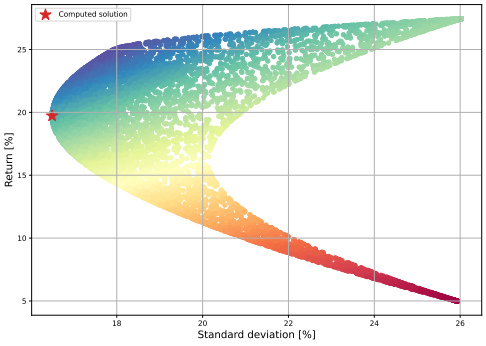

Figure 5: (a) 3 assets | Training set

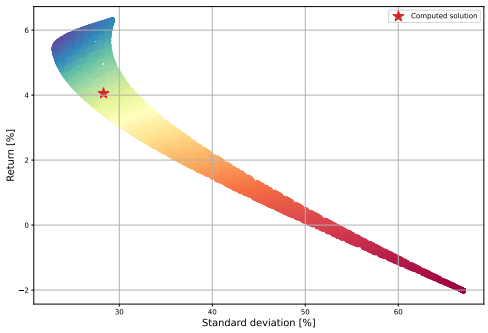

Figure 6: (b) 3 assets | Testing set

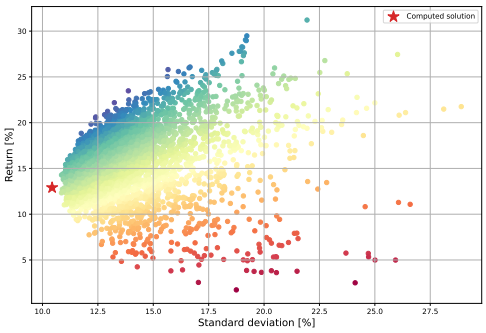

Figure 7: (c) 20 assets | Training set

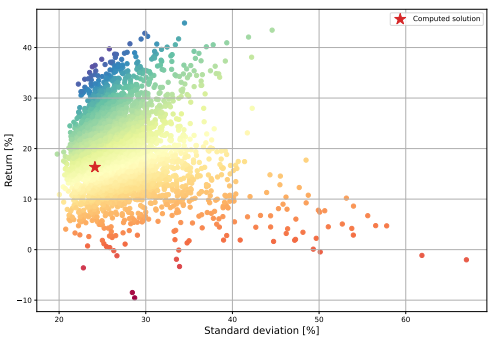

Figure 8: (d) 20 assets | Testing set

- **Validation of Optimality (Efficient Frontier):** In the training plots (Figures 5 and 7), the optimized Markowitz portfolio (red star) is located exactly on the upper-left boundary of the cloud. This empirically confirms that our solver successfully identified the **Pareto-optimal** solution: no random allocation offers a higher return for the same level of risk.
- **Impact of Diversification:** Comparing the two training scenarios illustrates the benefit of a larger investment universe.
  - Visual Shift: The entire cloud for $N = 20$ shifts significantly to the left. The minimum achievable standard deviation drops from approximately 18% ($N = 3$) to roughly 10% ($N = 20$).
  - Interpretation: This validates the theoretical principle that combining imperfectly correlated assets reduces unsystematic risk.
- **Out-of-Sample Robustness:** A major discrepancy appears on the testing data (Figures 6 and 8). The optimized portfolio, which was perfect on past data, falls **inside** the cloud on 2020 data. This indicates that the portfolio is no longer efficient out-of-sample. It highlights the main weakness of the Mean-Variance framework: it "overfits" historical correlations (2016-2019) which break down during a regime change (2020 crisis), leading to suboptimal realized performance.

3 Kurtosis Optimization

Question 10 Problem (5) is nonconvex because of the constraint $X = x\,x^T$ that is not convex.

4 Sequential Quadratic Programming (SQP)

4.1 A simple nonconvex problem

Question 11 We implemented the SQP method to solve the nonconvex problem defined in Equation (8). The core difficulty lies in the constraint $||x||_2^2 \geq 0.5$, which defines a nonconvex feasible set. At each iteration k, we linearized this constraint around the current point x^k. This transformation results in a sequence of convex Quadratic Programs that we solved using CVXPY.

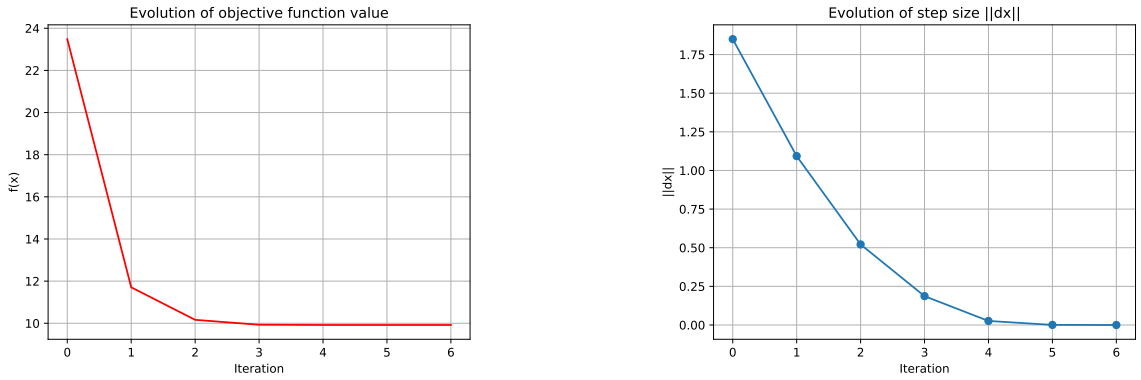

Figure 9: Convergence of SQP algorithm

The algorithm converges rapidly to a stationary point where the step size drops below the tolerance threshold.

We verified the feasibility of the final solution x^*: The box constraints $0 \leq x \leq 1$ are strictly respected. The nonconvex constraint $||x^*||_2^2 \geq 0.5$ is satisfied.

Question 12: Comparison with SciPy (SLSQP) We compared our custom SQP implementation against the standard `SLSQP` solver from `scipy.optimize`.

Table 1: Performance comparison: Custom SQP vs. SciPy SLSQP

Dimension	Metric	Custom SQP	SciPy SLSQP
$N = 20$	Final Objective	9.9188	9.9188
	Iterations	**6**	17
	Time (s)	0.071	**0.020**
$N = 200$	Final Objective	**115.60**	115.81
	Iterations	**10**	100 (Limit)
	Status	Converged	**Failed**

The analysis highlights two distinct behaviors:

- **Small Scale Efficiency ($N = 20$):** Both solvers converge to the exact same minimum. SciPy is faster in wall-clock time due to its optimized C-backend. However, our Custom SQP requires nearly 3× fewer iterations (6 vs 17), proving that solving the exact convex QP sub-problem yields a more efficient search direction than SciPy's.
- **Large Scale Robustness ($N = 200$):** As the dimension increases, the "exact sub-problem" approach proves superior. Our SQP successfully converged to a lower minimum in only 10 iterations, whereas SciPy failed to converge within the limit. This shows that our implementation is more robust for difficult non-convex problems.

4.2 Portfolio kurtosis problem

Question 13 We solved the nonconvex Kurtosis minimization problem using SCP. The algorithm iterates by solving convex sub-problems where the rank-1 constraint $X = xx^T$ is linearized.

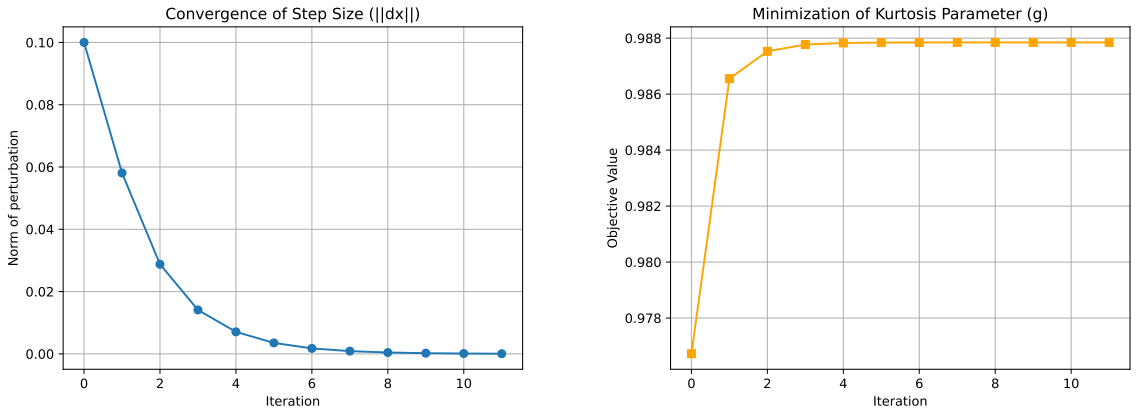

Figure 10: Analysis of SCP Convergence

Important: SCP is a local optimization method; it needs a feasible starting point so we solve the standard Minimum Variance problem to get x_0.

- **Robust Convergence:** The step size decreases linearly on a logarithmic scale and the objective function converges rapidly, indicating stable convergence. Note that the first step size is capped exactly at 0.10, confirming that our trust region constraint successfully prevented the solver from diverging due to invalid Taylor approximations in the early stages.

- **Portfolio Composition:** Unlike the Minimum Variance portfolio, the Min-Kurtosis solution favors specific defensive sectors. The top allocations are **PSA** (20.5%), **MMC** (15.1%), and **T** (10.3%), reflecting a strategy that minimizes extreme tail risks rather than just average volatility.

Question 14 We benchmarked our custom SCP implementation against the standard `SLSQP` solver.

Table 2: Comparison of solvers for Kurtosis Minimization ($N = 20$)

Metric	**Custom SCP (Q13)**	**SciPy SLSQP (Q14)**
Final Objective (g)	0.987846	0.987846
Difference	$\|x_{scp} - x_{slsqp}\| \approx 5.47 \times 10^{-8}$	
Iterations	12	12
Comp. Time	4.15 s	**0.033 s**

The results validate our SCP approach, as both solvers converged to the exact same local minimum. SciPy is significantly faster in wall-clock time as it uses an optimized compiled backend, whereas our custom method incurs overhead from parsing convex sub-problems in `CVXPY` at every step.

Question 15 We visualized the optimal portfolios against a Monte Carlo simulation of 1000 random portfolios generated on the testing data (2020-2021).

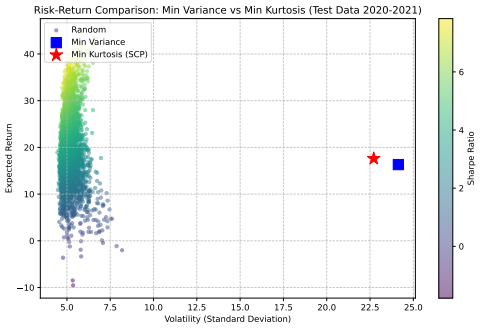

Figure 11: Resulting portfolios with Montecarlo

- **Regime Shift Impact:** Both the Minimum Variance (Blue Square) and Minimum Kurtosis (Red Star) portfolios lie to the right of the random cloud. This distance illustrates the "cost of stationarity": portfolios optimized on stable pre-COVID data (2016-2019) appear sub-optimal when projected onto the volatile post-COVID regime.

- **Performance Comparison:** Despite the difficult market conditions, the Minimum Kurtosis portfolio outperforms the Minimum Variance portfolio. It achieves a higher return ($\approx 18\%$ vs 16%) with slightly lower volatility. By explicitly minimizing tail risk (Kurtosis) rather than just variance, the SCP solver constructed a portfolio that proved more robust to the market crash.

Analysis of Allocation (Pie Charts): The Minimum Variance portfolio is heavily concentrated in low-volatility sectors. In contrast, the Minimum Kurtosis portfolio shows a distinct allocation logic, favoring assets like **PSA** and **MMC**, which likely exhibited lower tail-dependence during the training period.

5 Convex Relaxation

Question 16 Problem (10) is a semidefinite relaxation of Problem (5). The constraint $X = xx^T$ is relaxed to $X \succeq xx^T$ (via Schur complement). Therefore, the optimal value of (10) is a lower bound to (5).

Question 17 We solved the SDP relaxation using the `CLARABEL`/`SCS` solver.

- **Exact Relaxation (Tightness):** The solver returned a matrix X with **Rank 1** and a reconstruction error $\|X - xx^T\| \approx 0.0$. This indicates that the relaxation is **tight**: we "convexified" the problem without any approximation error.
- **Computational Trade-off:** While theoretically superior, the SDP method takes longer (1.58s) than the local SLSQP solver (0.03s).

6 Conclusions

So, we worked on portfolio optimization, using standard convex formulations (Markowitz) and then complex non-convex problems (Kurtosis minimization).

- **Algorithmic Trade-offs (Custom SCP vs. SciPy):** We implemented a SCP algorithm "from scratch" leveraging convex sub-problems. While the standard `SLSQP` solver of SciPy seemed faster looking at small-scale problems ($N = 20$), our custom SCP implementation demonstrated better robustness.
- **Mathematical Modeling of Non-Convexity:** By using first-order Taylor expansions and introducing perturbation variables, we were able to transform a hard non-convex problem into a sequence of easier subproblems.
- **Financial Robustness and Regime Shifts:** Going from the training set (2016-2019) to the testing set (2020-2021) showed the limits of historical optimization. The standard Mean-Variance portfolio suffered from the "stationarity gap" during the COVID crash. And our Minimum Kurtosis portfolio outperformed it on the test set. This confirms that minimizing "fat tail" risks provides a crucial safety net during extreme market regimes such as the one we faced in 2020.

Bonus Machine learning could be used to predict μ and Σ more accurately (e.g., using LSTM or Transformers on time series), but it is difficult because financial data is extremely noisy (low signal-to-noise ratio) and non-stationary.